1962

The design of a single failure detecting translator

David Robert Hughes
Lehigh University

THE DESIGN OF A SINGLE FAILURE DETECTING TRANSLATOR

by

David Robert Hughes

A THESIS

Presented to the Graduate Faculty

of Lehigh University

in Candidacy for the Degree of

Master of Science in Electrical Engineering

Lehigh University

1962

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

May 16, 1962

(Date)

John Karakash

Professor in Charge

John Karakash

Head of the Department

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Professor John J. Karakash for his encouragement and guidance throughout the author's graduate and undergraduate study. He is also indebted to Lehigh University for this opportunity for advance study through an International Telephone and Telegraph Company Fellowship. The author also wishes to acknowledge the assistance of Mr. William Berry for his work in coordinating the completion of this thesis during the author's employment at the International Business Machines Corporation. Gratitude is also expressed to Miss Dorothy Parker for the typing of this thesis.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Cont'd.)

1 INTRODUCTION

The complex computers being developed for today's commercial and scientific world are capable of doing many different kinds of operations, are made up of many different pieces of equipment, use many different codes. A typical system is made up of such items as a memory, a central processing unit, a console, input-output synchronizers or multiplexers, and many different forms of input-output equipment, such as card readers, card punches, printers, typewriters, magnetic tape units, paper tape units, etc. Each of these units, in turn, is made up of many smaller units, such as adders, registers, control rings, translators, etc. Breaking these units down even further discloses that they are composed of logic circuit blocks which comprise "and" circuits (circuits which have a certain output level when all the inputs are at a certain level), "or" circuits (circuits which have a certain output level if any one or more of several inputs are at a certain level), and inverters (circuits which have an output level which is opposite to their input level); storage circuits, such as triggers and latches; and power or driving circuits, such as power inverters and emitter followers.

In this study, a translator which will translate from a two-out-of-five code to a binary coded decimal code will be designed using the last set of building blocks described -- logic blocks. With the requirements of new computers shifting to in-line or constant processing

rather than batch processing (running one job at a time), the reliability of a computer is of prime concern. The computer will, in turn, be no more reliable than its component parts (translators, registers, etc.). Since circuits or logic blocks have not yet been developed which cannot fail, the component parts must be designed so that if a circuit does fail, an error must be flagged. In this manner, the reliability of the system will be increased with each component not only doing its operation correctly but checking its own circuits also. In this study, the translator will be designed in a straightforward manner using conventional methods. It will then be analyzed to discover the effects of circuit failures. The translator will then be redesigned, if necessary, to provide failsafe operation.

2 CODES USED IN THE TRANSLATOR

The input to the translator designed in this report will be a two-out-of-five code. This code represents every digit by the presence of two out of five bits and absence of the other three out of five bits. A two-out-of-five code is capable of representing only ten digits; and this code represents the digits zero through nine, using a zero, a one, a two, a three, and a six bit. The structure of the code is such that the sum of the two bits designating a particular digit is equal to that digit with the exception of zero. For example, the digit one is represented by the presence of a zero bit and a one bit. Two is represented by the presence

of a zero bit and a two bit. In the same manner, all ten digits from one to nine are represented. Zero is represented by a one bit and a two bit. Table I gives the digits from zero through nine and the bit structure which is used to represent these digits in the two-out-of-five code. The one shown in a column of bits represents the presence of that bit for that particular digit. The zero represents the absence of a bit.

Two-Out-of-Five Code

Decimal Digit	Two-Out-of-Five Representation				
	0 Bit	1 Bit	2 Bit	3 Bit	6 Bit
0	0	1	1	0	0
1	1	1	0	0	0
2	1	0	1	0	0
3	1	0	0	1	0
4	0	1	0	1	0
5	0	0	1	1	0
6	1	0	0	0	1
7	0	1	0	0	1
8	0	0	1	0	1
9	0	0	0	1	1

Table I

One of the advantages of the two-out-of-five code is the ability to easily check the validity of a digit representation. Since a number is represented in only two of the five bits, the checking circuit need only check for the presence of only two bits. If the number contains less than two or more than two bits, it is invalid and an error is flagged. Codes such as the binary decimal code do not have this feature because the number of bits representing a character is not constant. Only the parity can be checked on such a code. (Does it contain an odd number or even number of digits?)

One of the big disadvantages of the two-out-of-five code is that only ten digits can be represented by the code. If alphanumeric information is to be represented, two digits are required to represent the alphanumeric character. For instance, if one wants to represent the letter A in a two-out-of-five code, one would have to have some kind of a number to represent the first ten letters of the alphabet, and also one would have to have some kind of number to represent what position in the first ten characters in the alphabet A occupies. For instance, if the letter A were to be represented, the first digit could be a one (1) since A is in the first ten letters of the alphabet. The second digit would also be a one since A is the first letter in this group. Eleven would then be used to represent an A. The next grouping would start with K. This grouping would be represented by the digit two (2). A K would then be represented by the number twenty-one.

Since the two-out-of-five code requires two storage locations (one for each digit) to represent alphanumeric information, the code is usually used where only numeric information is processed. Its high reliability and the ease with which it is checked make it very useful in an area like the addressing portion of a computer.

The output of the translator will be in the binary coded decimal code. In this code, the digits from one through nine are represented by their binary equivalents. For example, a seven is represented by the number 0111 where the zero represents an eight; the first one, a four; the second one, a two; and the third one, a one. The digit represented is the sum of the ones; in this case, 4 + 2 + 1 is seven. The zero digit in this code is represented as a ten would be; that is, by an eight bit and a two bit (1010). Since the number of bits used to represent a digit varies (a one is 0001, a six is 0110, a seven is 0111), the validity of a BCD combination cannot be determined by looking for a certain number of bits. A fifth bit, called a C bit or a check bit, is added. This bit is added to each character having an odd number of bits if the user of the code decides to use even parity (even number of bits) for his valid characters. If it is decided that the parity of the characters should be odd, the C bit is added to all digits having an even number of bits present.

For this application in the translator, odd parity has been selected to be maintained in the BCD output. An advantage of the odd parity is that no two digits can be merged together and result in a valid

character. This is an advantage in that if a register in a computer is being set from two different sources at the same time, an error should be flagged. The following example will show how the merger of two digits can be detected if odd parity is used:

Even Parity

C Bit	8	4	2	1							
0	0	0	1	1		three		1	0	0	1	1
1	0	1	0	0		four		0	0	1	0	0
1	0	1	1	1	valid	seven	invalid	1	0	1	1	1

The main advantage of the binary coded decimal code is that it closely resembles the Hollerith code widely used in punched cards. Since many of the input-output information into the computer comes from punched cards, much of the input to and output from a computer is made up of this binary coded decimal code. The BCD code can also easily be used to represent alphanumeric information since it uses all sixty-four combinations of the six bits in it (A bit, B bit, 8 bit, 4 bit, 2 bit, and 1 bit). Only one location in storage is required to represent such a character.

Table II shows the digits zero through nine and their BCD representations. Since only numeric information is shown in this illustration, the A bit and the B bit of the BCD code are not shown since they are not present for numeric information.

Binary Coded Decimal Code

Decimal Digit	Binary Coded Decimal Representation C Bit	8 Bit	4 Bit	2 Bit	1 Bit
0	1	1	0	1	0
1	0	0	0	0	1
2	0	0	0	1	0
3	1	0	0	1	1
4	0	0	1	0	0
5	1	0	1	0	1
6	1	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	1	1	0	0	1

Table II

Many computers use the binary coded decimal code for their data flow because much of their input and output information is already in this code. This means that the storage area or memory of these computers would probably store its information in the BCD code. Since the programs of modern computers are also stored in memory, the addresses in these programs will have to be converted from BCD to a two-out-of-five code to address the storage unit if a two-out-of-five code is chosen for this purpose. Also, addresses which are stored

must be converted in BCD code. Therefore, it can be seen that an application exists for translators from one code to another.

3 TRANSLATOR SPECIFICATIONS

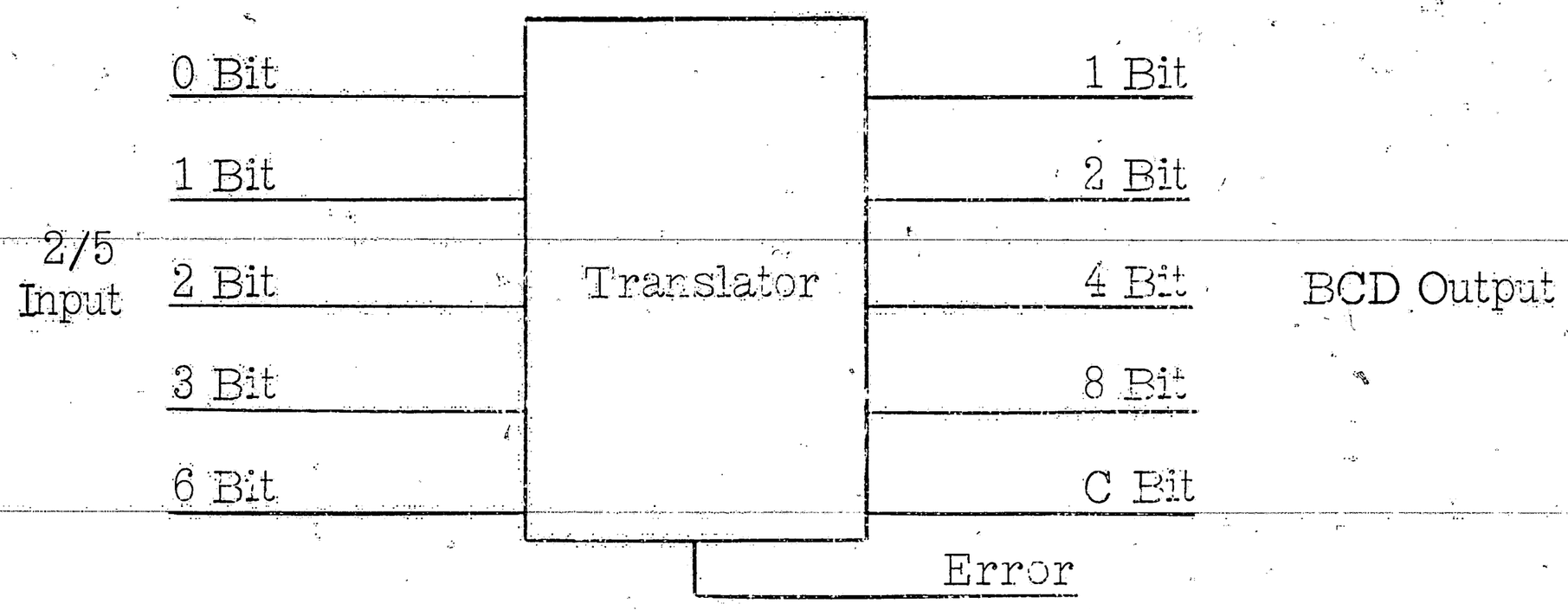
The circuit system that is being used in the design of this translator is the NOR circuit system, a resistor-transistor logic group. Its signal levels swing from an up level of zero volts to a down level of minus seven to minus twelve volts. We shall represent the presence of a bit at the input of our translator as the up level or zero volt signal. The input to it shall have one line for each bit in the two-out-of-five code with sufficient current available to drive all the logic circuits required by the translator. A valid combination of the two-out-of-five code consists of two of the five input lines being positive (zero volts) and the other three being negative. It shall be assumed for the purpose of this study that the translator has to accept either valid or invalid information.

The output requirements of the translator also require the presence of a bit be represented by the up level. There will be five lines out -- one line to represent a one bit, one to represent a two bit, one to represent a four bit, one to represent an eight bit, and one to represent a C bit which will be used to determine parity. No drive requirements will be specified on the output. The output must be a valid BCD code combination representing any digit from zero

through nine, providing that the input to the translator was a valid two-out-of-five representation of the same digit. An error must be flagged under any one of the following circumstances:

1. The input to the translator is an invalid two-out-of-five code combination (less than two bits or more than two bits).
2. The output character has even parity.
3. The output character is other than a digit from zero through nine.
4. The output digit is different from the input digit.

The translator as defined above will then have the form shown in Figure 1.



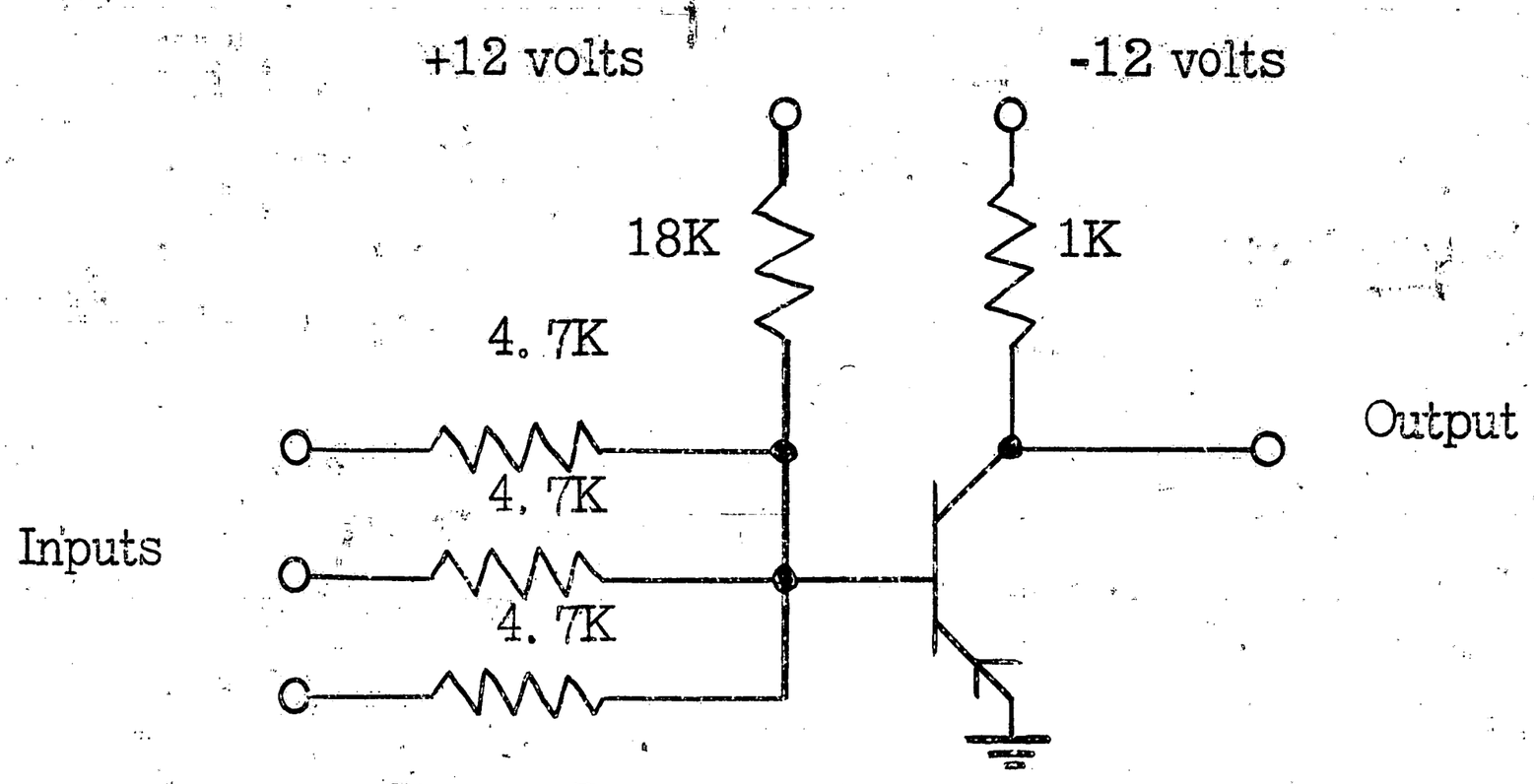
Translator Data Flow
Figure 1

4 CIRCUIT FAMILY TO BE INCORPORATED INTO THE DESIGN

Before beginning the design of the translator, the NOR circuit family will be briefly described. The name NOR means negative "or"; that is, if any one of the inputs to a logic block assumes the negative or down level, a positive output occurs. The logic is accomplished by placing a resistor divider network on the base of a PNP transistor.

There are many advantages to using the NOR circuit family. One of the main advantages of this family is that the logic block is logically complete. That is, three logical functions of "and," "or," and inversion can be obtained by using the NOR block. The NOR block gives a minus "or" function or a plus "and" function. Some other advantages are its low cost, its good reliability, and the fact that the signal lines vary only between two voltages. The disadvantages of this circuit family lie mainly in the fact that only the plus "and" and minus "or" functions are available. The plus "or" and the minus "and" functions are not available. This requires the use of inverters to get positive lines to a negative input level to perform the "or" function, and also to convert negative lines to positive inputs to perform the "and" function. Another disadvantage of this system sometimes found in other logic families is that the complement functions cannot be generated by tying blocks together at the collectors of the transistors.

The blocks that will be used in the design of the translator will now be described briefly. The first block that will be looked at is the logic block. This block has the function of a plus "and" or a minus "or." The circuit diagram for this block is shown in Figure 2.

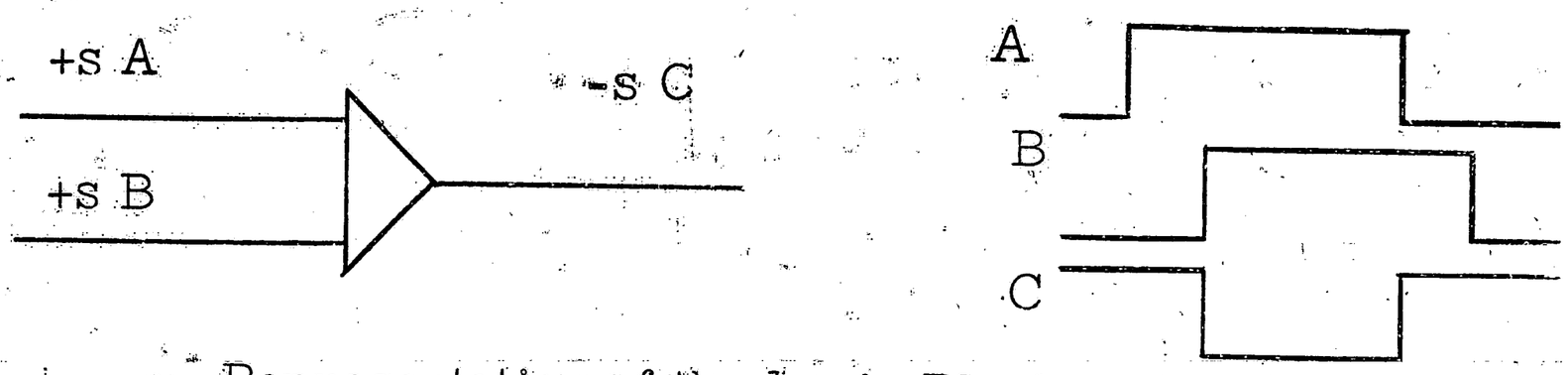


Circuit Diagram of the Logic Block

Figure 2

This plus "and" or minus "or" may have two or three inputs. The PNP transistor used in this circuit requires a negative input (with respect to ground) to the base to turn it on. (The "on" state is when the transistor is conducting in the saturation region.) A positive input to the base with respect to ground will turn the transistor off. (The "off" state is when the transistor is not conducting.) If each one of the inputs to this were held at ground or zero volts, the base of the transistor would be positive with respect to the emitter which is grounded, because the voltage divider on the base would place a positive voltage of about two volts on the base. Therefore, with each

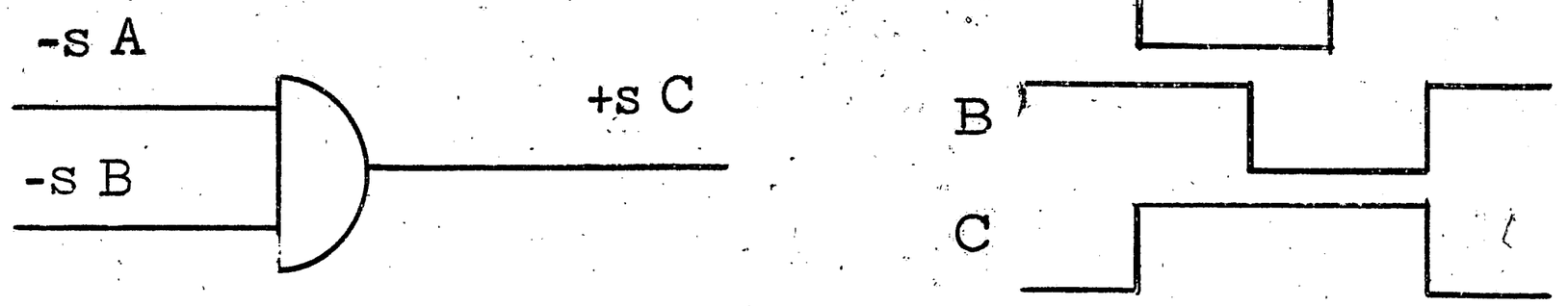
of the inputs at ground, the transistor will be off and the output level will be between seven and twelve volts negative, depending on the current being drawn by the load on this block (additional blocks connected to the output). The block is represented as shown in Figure 3, and the signals associated with it are shown.



Representation of the Logic Block as an And Circuit

Figure 3

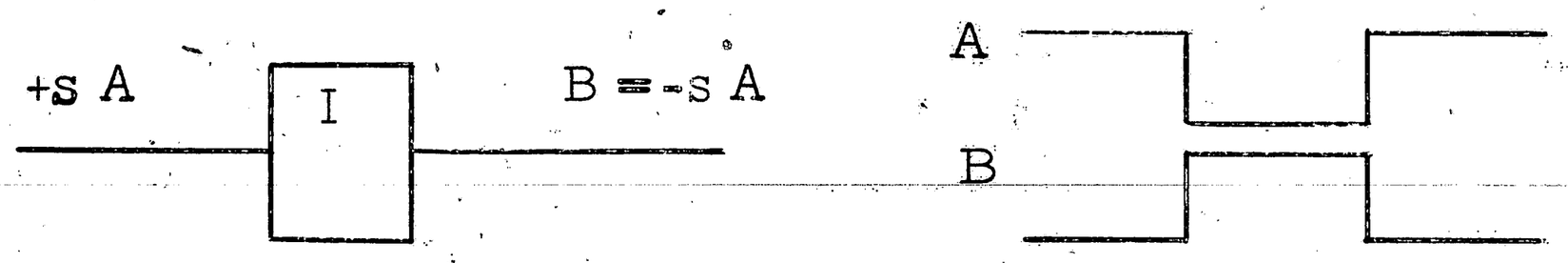
If one of the inputs to the logic block is at a minus level, the voltage divider will place a voltage at the base which will then be negative with respect to ground. This will cause the base to be at a lower potential than the emitter and the transistor will conduct. The output signal level at the collector will then be approximately zero volts since the emitter-collector voltage drop is almost zero. This situation then represents minus "or" function. One of the inputs at the minus level then causes the output to rise to the positive level. The block representation and signals are shown in Figure 4.



Representation of the Logic Block as an "Or" Circuit

Figure 4

When the logic block is used as an inverter by using only one input, the block is then represented as shown in Figure 5.



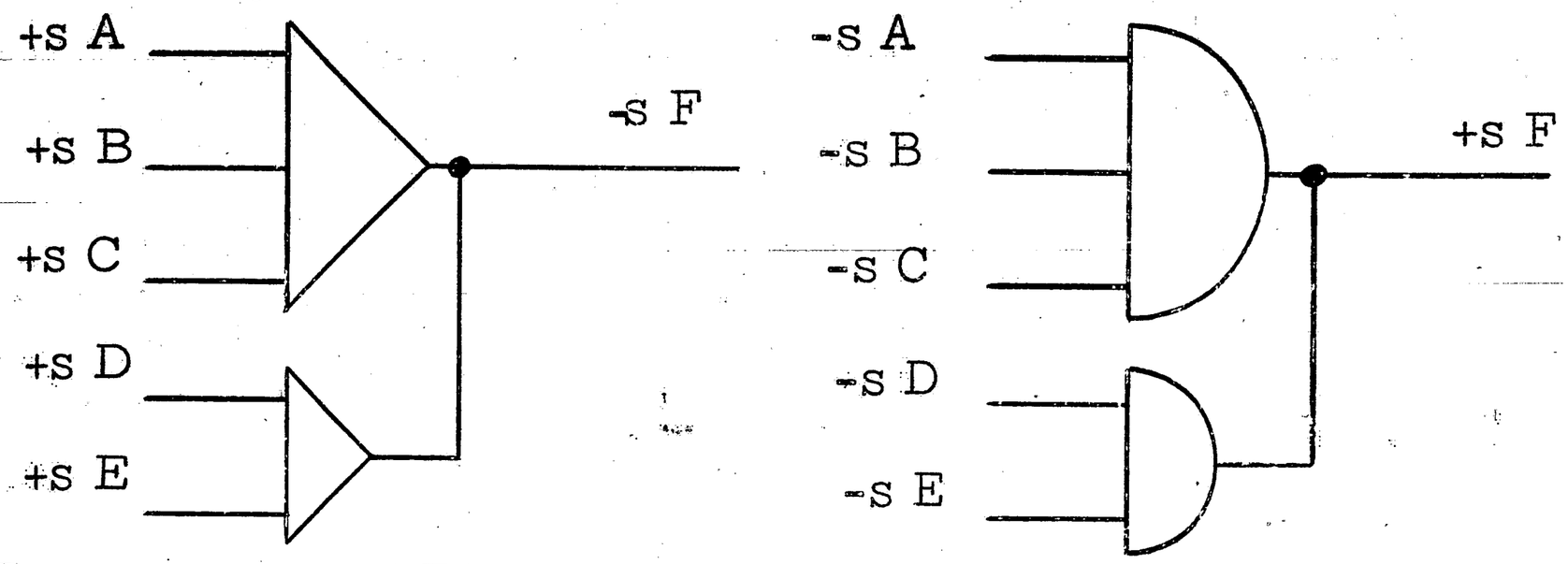
Representation of the Logic Block as an Inverter

Figure 5

In the above diagrams, the "S" in front of the signal name indicates that the voltage swing is between zero volts and minus twelve volts. The plus sign (+) means that the signal is present when that line is positive (zero volts). The minus sign (-) means that the signal is present when the line is negative.

The logic functions of the NOR logic block may be extended to more than three inputs by having two or more transistors share the same collector load. If any one transistor conduits because

one of its inputs became negative, the output of all the transistors would be zero volts. If no inputs were negative, the output would remain at the negative level. Expanded "and" and "or" functions are shown in Figure 6.



Representation of Expanded Functions Using Logic Blocks

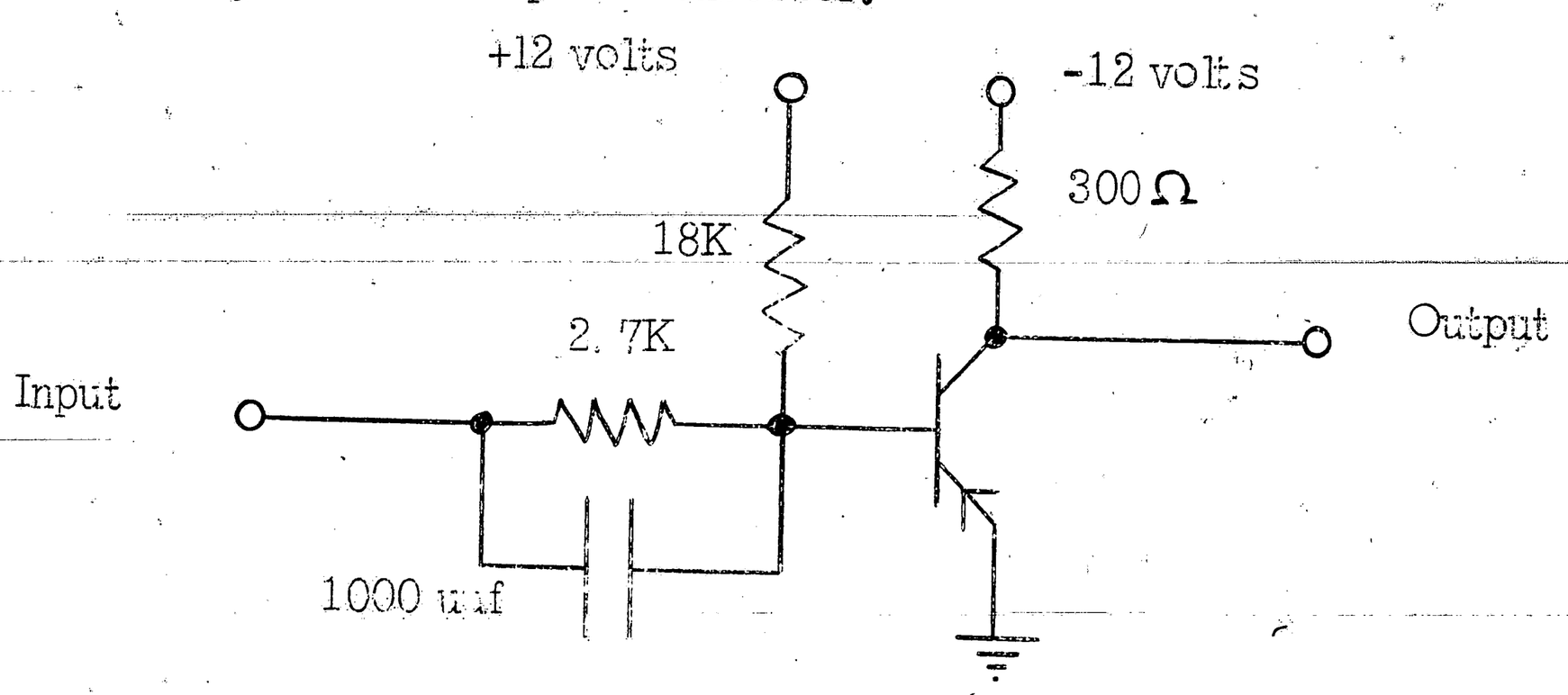
Figure 6

By tying these logic blocks together at their collectors, a five-way "and" circuit can be formed by using a two-way block and a three-way, an eight-way "or" can be formed by using two three-ways and a two-way, etc.

For analysis of the translator, it will be assumed that the logic block, when it fails, will fail in one of two ways. In one instance it will be assumed that it will fail because the transistor in the circuit has shorted (zero ohms resistance from emitter to collector). In this case the output of the logic block will always be at the up level (zero volts). An "or" circuit will always be satisfied whether one of the inputs is negative or not, and an "and" circuit will never be satisfied even if all its inputs are positive. The second

failure case occurs when the transistor opens; that is, the emitter to collector or base to collector resistance is infinite. In this situation, the output will always be at the down level. An "and" circuit will always be satisfied regardless of its inputs; and an "or" circuit will never be satisfied even if any or all inputs are negative.

The next member of the NOR circuit family to be discussed is the power inverter. The purpose of this block is to invert the signal and give it enough drive to drive ten bases (inputs to logic blocks). The logic block drives only three bases. Figure 7 is the circuit diagram for the power inverter.

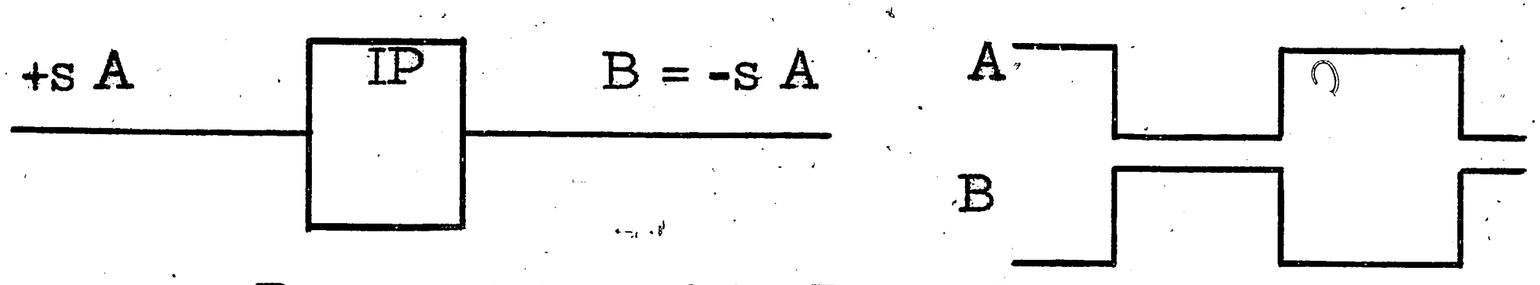


Circuit Diagram for the Power Inverter

Figure 7

When a signal of zero volts is presented at the input of the block, the base is held positive with respect to ground and the emitter by the resistor divider network on the base. The PNP transistor is turned off and a negative output is represented, the level being determined by the amount of current the load draws through the load resistor. If

the input goes to minus seven to minus twelve volts, the base becomes negative with respect to the emitter and the transistor conducts with the output rising to approximately zero volts. The capacitor across the input resistor gives an additional shot of current initially to the base so that the transistor will turn on more quickly. The representation for a power inverter is shown in Figure 8.

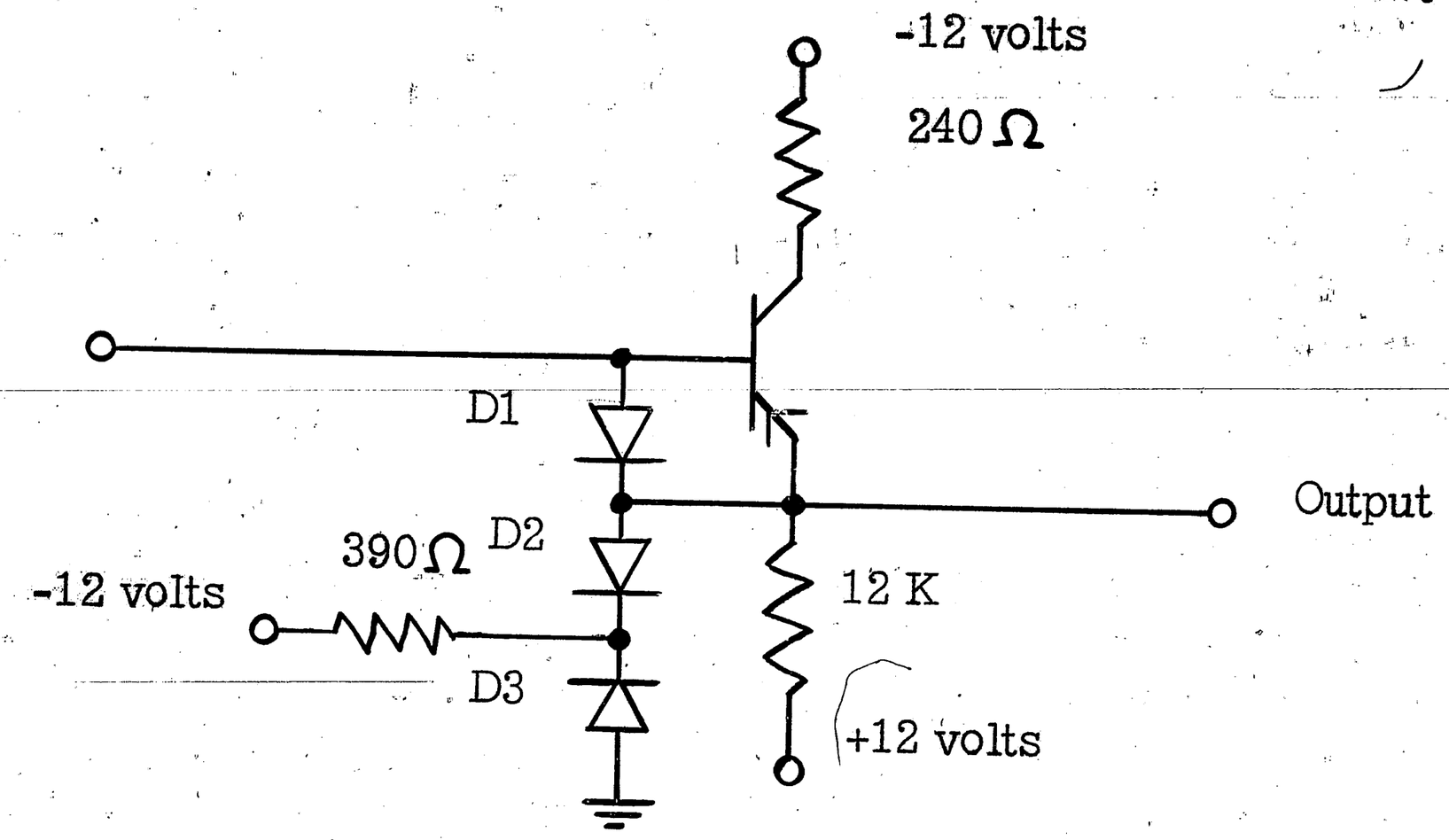


Representation of the Power Inverter

Figure 8

For later analysis, this circuit will also be considered to fail in two ways. If the transistor shorts, the output will always be at zero volts. If the transistor opens, the output will always be negative.

The last circuit in the NOR family which will be used in the translator is the emitter follower. This circuit has an input level swing from zero volts to minus seven to minus twelve volts and gives an output of the same voltage level but with enough current to drive ten bases. The circuit diagram for this circuit is shown in Figure 9.



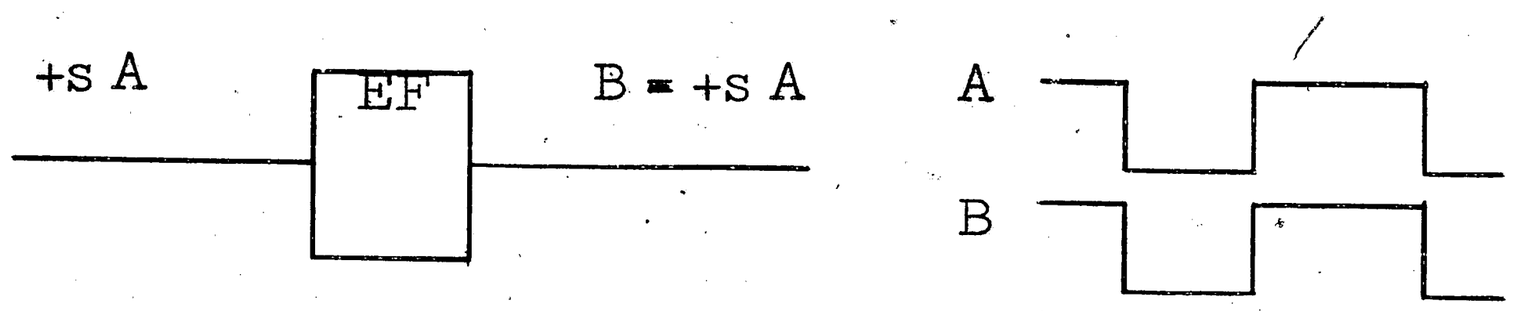
Circuit Diagram for the Emitter Follower

Figure 9

The PNP transistor is biased in such a manner that the output level is positive or zero volts for an input level of zero volts. The three diodes are used to develop a clean output level when the input is at ground. When the input is negative, the transistor will conduct and the output or emitter will follow the base. If the transistor in this circuit opens, the output will remain at the ground level. The diodes will hold this level without regard for the input level. If the transistor shorts, the output will be negative and the diodes will have no affect. If diode D1 shorts, the emitter follower will not be in the circuit because the transistor will be bypassed. The down level will not go negative enough because the load resistor of the circuit which is driving the emitter follower will be required to deliver too much current. The output will appear to be near ground. If diode D1

opens, the levels will not be affected; but the turn on time of the transistor may be increased if the base is back biased beyond the forward drop of D1. A short of diode D2 or an open of diode D3 would have the effect of placing the emitter of the transistor at minus twelve volts and giving the emitter follower a negative output all the time. An open of D2 or a short of D3 would cause a shift in the up level of the output. The short of D3 would cause the emitter follower to increase the back bias of the bases of the blocks it is driving and thus increase their turn on delays. The open of D2 may allow the output to drop below the minimum up level and thus cause the output never to reach the up level. The voltage drop of the base to emitter junction would be greater than the voltage which results as the difference between the forward drops of D2 and D3.

Figure 10 shows the block representation for the emitter follower.



Representation of the Emitter Follower

Figure 10

5 CONVENTIONAL DESIGN OF THE TRANSLATOR

Having described the circuit family to be used and having defined the requirements of the translator to be designed, the de-

sign is now undertaken using conventional methods. The first step in the layout of the network is to devise a table which gives the coded values for each digit to be translated. Table III contains in the first column the digit to be translated, in the second column is the two-out-of-five representation for this digit, and in the third column is the binary-coded-decimal representation. (Binary-coded-decimal will be referred to as BCD from now on in this paper.) In this chart the presence of a bit is represented by a "1." The absence of the bit is represented by a "0."

Digit Encoding

	Two Out of Five Representation					BCD Representation				
Digit	0	1	2	3	6	1	2	4	8	C
0	0	1	1	0	0	0	1	0	1	1
1	1	1	0	0	0	1	0	0	0	0
2	1	0	1	0	0	0	1	0	0	0
3	1	0	0	1	0	1	1	0	0	1
4	0	1	0	1	0	0	0	1	0	0
5	0	0	1	1	0	1	0	1	0	1
6	1	0	0	0	1	0	1	1	0	1
7	0	1	0	0	1	1	1	1	0	0
8	0	0	1	0	1	0	0	0	1	0
9	0	0	0	1	1	1	0	0	1	1

Table III

Using Table III, the following equations can be written for each BCD bit. The equations are written for these bits since these will be the output lines of the translator. These equations are written using the digits as terms.

1 Bit = 1 + 3 + 5 + 7 + 9

2 Bit = 0 + 2 + 3 + 6 + 7

4 Bit = 4 + 5 + 6 + 7

8 Bit = 0 + 8 + 9

C Bit = 0 + 3 + 5 + 6 + 9

In the above equations, the plus sign (+) is used to represent the logical function "or." This will hold true in all equations written. The "and" function will be written as a dot (•). Thus, A + B means A or B; A • B means A and B. A line drawn over a term means the absence of that term. $\overline{A}$ means "not" A.

5.1 DESIGN BY BOOLEAN REDUCTION

If the two-out-of-five code values are substituted for each digit in the above expression for the one bit, an equation will be obtained giving the output line in terms of the input lines.

$$1\text{ Bit} = 0 \cdot 1 \cdot \overline{2} \cdot \overline{3} \cdot \overline{6} + 0 \cdot \overline{1} \cdot \overline{2} \cdot 3 \cdot \overline{6} + \overline{0} \cdot \overline{1} \cdot 2 \cdot 3 \cdot \overline{6} + \overline{0} \cdot 1 \cdot \overline{2} \cdot \overline{3} \cdot 6 + \overline{0} \cdot \overline{1} \cdot \overline{2} \cdot 3 \cdot 6$$

If this expression were to be implemented directly, it would need two transistors per digit since a two-way logic block and a three-way logic block would have to be collector "and"ed to form a five-way and. Two transistors would also be used to form the five-way

"or" circuit for the output. A total of twelve transistors would be required to develop a line which is at ground level if the one bit is to be present.

Equations can often be reduced by collecting the common terms together. In this equation it can be seen that there is no more than one term that is common to a group of terms. Since the term 3 appears three times in the expression, let us factor it out and see what happens to the equation.

$$1\ \text{Bit} = 3(0\cdot\overline{1}\cdot\overline{2}\cdot\overline{6} + \overline{0}\cdot\overline{1}\cdot 2\cdot\overline{6} + \overline{0}\cdot\overline{1}\cdot\overline{2}\cdot 6) + \overline{3}(0\cdot 1\cdot\overline{2}\cdot\overline{6} + \overline{0}\cdot 1\cdot\overline{2}\cdot 6)$$

This expression can now be reduced further by collecting more common terms.

$$1\ \text{Bit} = 3\left[\overline{1}\ (0\cdot\overline{2}\cdot\overline{6} + \overline{0}\cdot 2\cdot\overline{6} + \overline{0}\cdot\overline{2}\cdot 6)\right] + \overline{3}\left[1(0\cdot\overline{2}\cdot\overline{6} + \overline{0}\cdot 1\cdot\overline{2}\cdot 6)\right]$$

$$1\ \text{Bit} = 3\cdot\overline{1}\ (0\cdot\overline{2}\cdot\overline{6} + \overline{0}\cdot 2\cdot 6 + \overline{0}\cdot\overline{2}\cdot 6) + \overline{3}\cdot 1\,(0\cdot\overline{2}\cdot\overline{6} + \overline{0}\cdot 2\cdot\overline{6})$$

Eight transistors are needed to implement this expression as shown in Figure 11

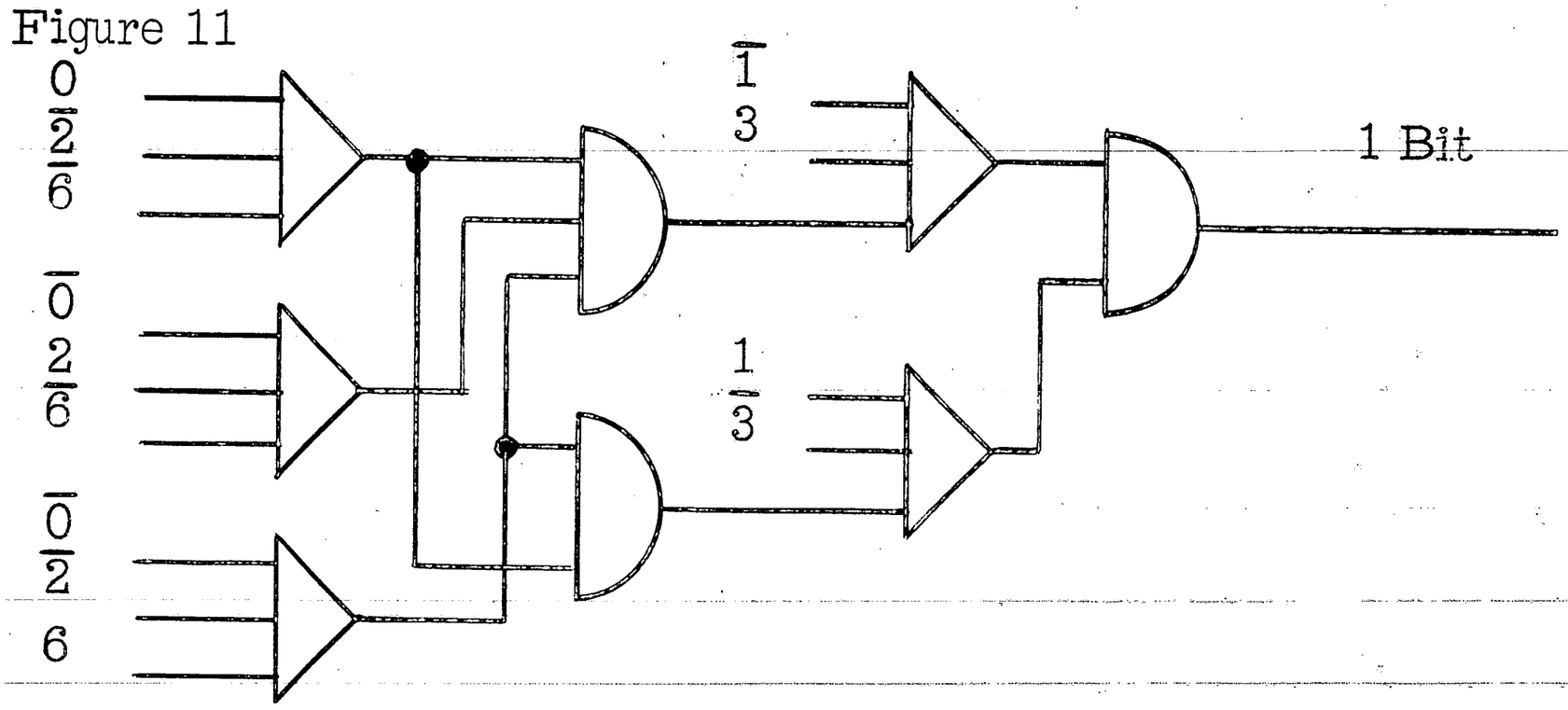


Figure 11

Implementation of the Expression for a One Bit

The other four output lines are developed in the same manner.

$$2\text{ Bit} = \overline{0}\cdot 1\cdot 2\cdot \overline{3}\cdot \overline{6} + 0\cdot \overline{1}\cdot 2\cdot \overline{3}\cdot \overline{6} + 0\cdot \overline{1}\cdot \overline{2}\cdot 3\cdot \overline{6} + 0\cdot \overline{1}\cdot \overline{2}\cdot \overline{3}\cdot 6 + \overline{0}\cdot 1\cdot \overline{2}\cdot \overline{3}\cdot 6$$

$$= 0(\overline{1}\cdot 2\cdot \overline{3}\cdot \overline{6} + \overline{1}\cdot \overline{2}\cdot 3\cdot \overline{6} + \overline{1}\cdot \overline{2}\cdot \overline{3}\cdot 6) + \overline{0}(1\cdot 2\cdot \overline{3}\cdot \overline{6} + 1\cdot \overline{2}\cdot \overline{3}\cdot 6)$$

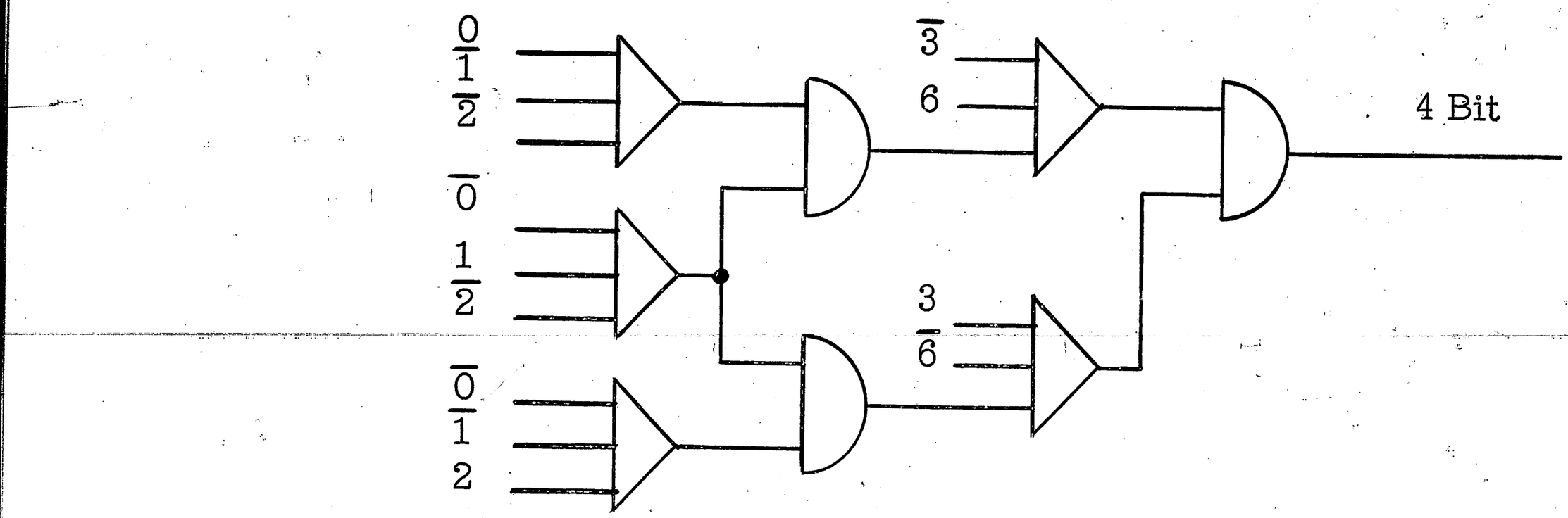
$$= 0\cdot \overline{1}(2\cdot \overline{3}\cdot \overline{6} + \overline{2}\cdot 3\cdot \overline{6} + \overline{2}\cdot \overline{3}\cdot 6) + \overline{0}\cdot 1(2\cdot \overline{3}\cdot \overline{6} + \overline{2}\cdot \overline{3}\cdot 6)$$

Eight transistors are required to implement this expression.

$$4\text{ Bit} = \overline{0}\cdot 1\cdot \overline{2}\cdot 3\cdot \overline{6} + \overline{0}\cdot \overline{1}\cdot 2\cdot 3\cdot \overline{6} + 0\cdot \overline{1}\cdot \overline{2}\cdot \overline{3}\cdot 6 + \overline{0}\cdot 1\cdot \overline{2}\cdot \overline{3}\cdot 6$$

$$= \overline{2}\cdot \overline{3}\cdot 6(0\cdot \overline{1} + \overline{0}\cdot 1) + 3\cdot \overline{6}(\overline{0}\cdot 1\cdot \overline{2} + \overline{0}\cdot \overline{1}\cdot 2)$$

$$= \overline{3}\cdot 6(0\cdot \overline{1}\cdot \overline{2} + \overline{0}\cdot 1\cdot \overline{2}) + 3\cdot \overline{6}(\overline{0}\cdot 1\cdot \overline{2} + \overline{0}\cdot \overline{1}\cdot 2)$$



Implementation of the Expression for a Four Bit

Figure 12

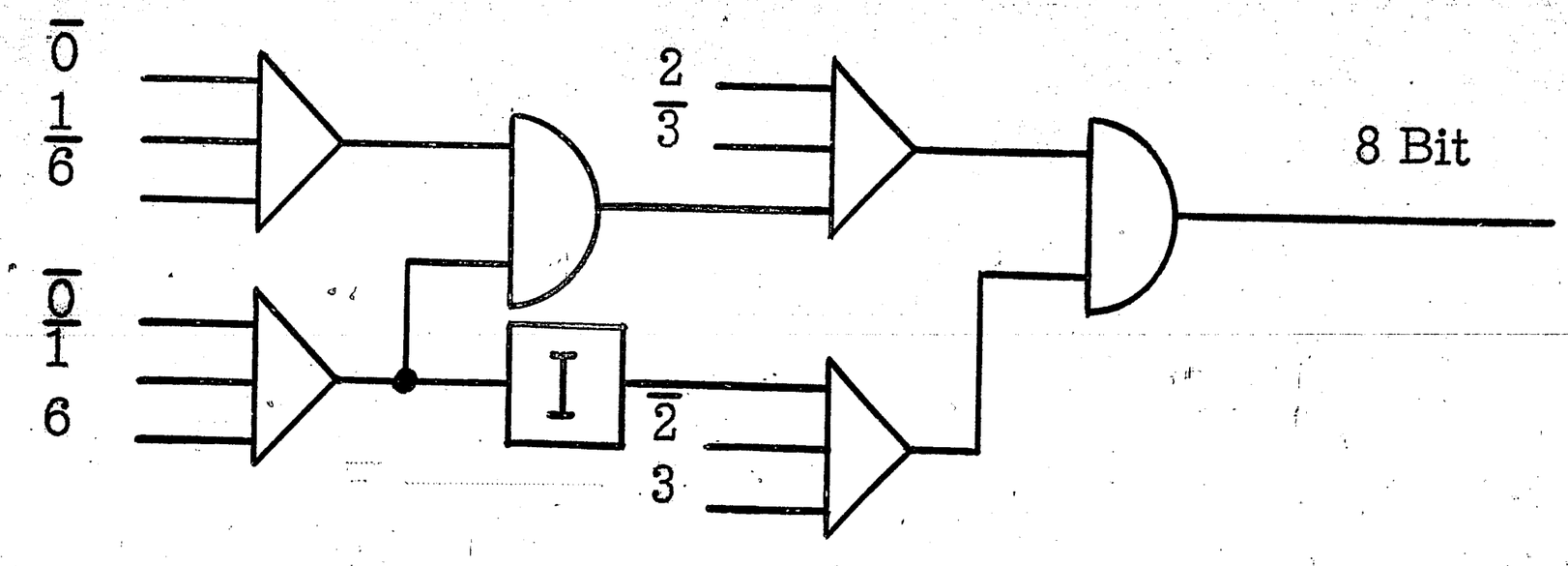
Eight transistors are required for the four bit.

$$8\text{ Bit} = \overline{0}\cdot 1\cdot 2\cdot \overline{3}\cdot \overline{6} + \overline{0}\cdot \overline{1}\cdot 2\cdot \overline{3}\cdot 6 + \overline{0}\cdot \overline{1}\cdot \overline{2}\cdot 3\cdot 6$$

$$= \overline{0}(1\cdot 2\cdot \overline{3}\cdot \overline{6} + \overline{1}\cdot 2\cdot \overline{3}\cdot 6 + \overline{1}\cdot \overline{2}\cdot 3\cdot 6)$$

$$= \overline{0}\ 2\cdot \overline{3}(1\cdot \overline{6} + \overline{1}\cdot 6) + \overline{1}\cdot \overline{2}\cdot 3\cdot 6$$

$$= 2\cdot \overline{3}(\overline{0}\cdot 1\cdot \overline{6} + \overline{0}\cdot \overline{1}\cdot 6) + \overline{0}\cdot \overline{1}\cdot \overline{2}\cdot 3\cdot 6$$



Implementation of the Expression for an Eight Bit

Figure 13

Seven transistors are required to implement the eight bit

$$C\ Bit = \overline{0}\cdot 1\cdot 2\cdot \overline{3}\cdot \overline{6} + 0\cdot \overline{1}\cdot \overline{2}\cdot 3\cdot \overline{6} + \overline{0}\cdot \overline{1}\cdot 2\cdot 3\cdot \overline{6} + 0\cdot \overline{1}\cdot \overline{2}\cdot \overline{3}\cdot 6 + \overline{0}\cdot \overline{1}\cdot \overline{2}\cdot 3\cdot 6$$

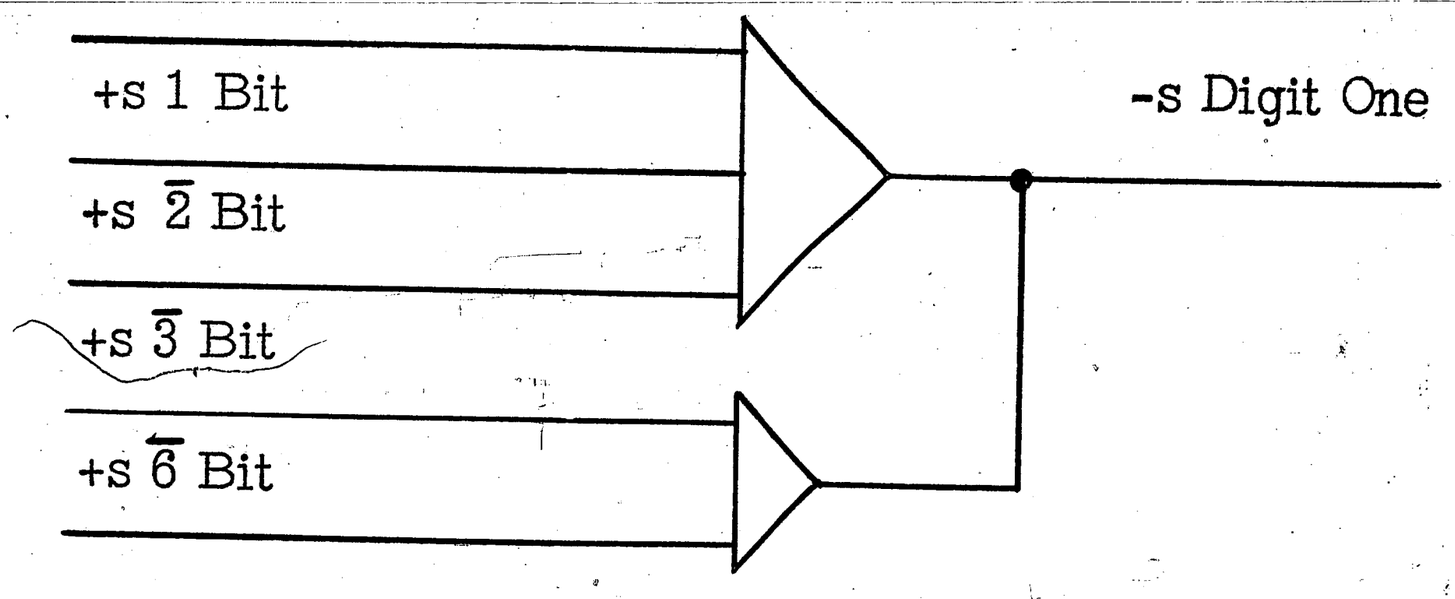
$$= \overline{1}\cdot \overline{2}(0\cdot 3\cdot \overline{6} + 0\cdot \overline{3}\cdot 6 + \overline{0}\cdot 3\cdot 6) + 2\cdot \overline{0}(1\cdot \overline{3}\cdot \overline{6} + 1\cdot 3\cdot \overline{6})$$

Ten transistors are required to implement this check bit.

If the translator is to be built from the implementation of these equations, a total of forty-one transistors would be required.

5.2 DESIGN BY DIGIT IMPLEMENTATION

An alternate method that can be used to design the translator is to develop all of the ten decimal digits in terms of their two-out-of-five bits. This would require two transistors for each bit or a total of twenty for all the digits. As an example, the digit one is shown in Figure 14.

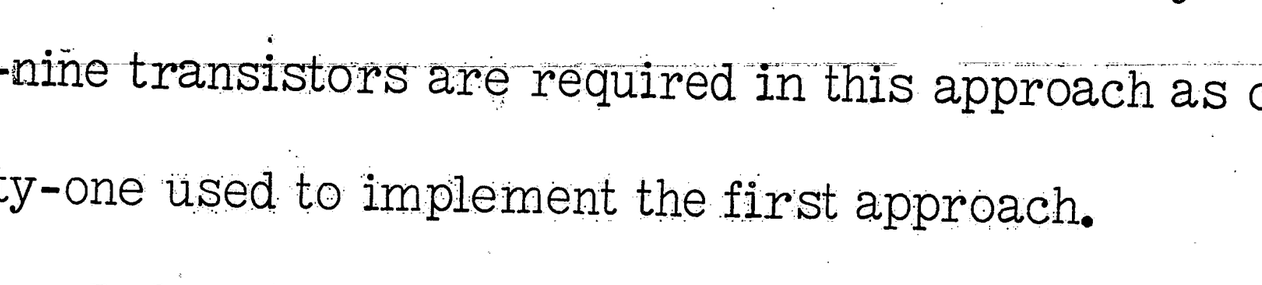


Implementation of a Digit

Figure 14

If the expressions which contain the output lines of the translator as a function of the digits are used to lay out the logic, the translator will take on the form shown in Figure 15. This layout requires nine transistors to form the BCD bits from the digits. A total of twenty-nine transistors are required in this approach as compared to the forty-one used to implement the first approach.

In both implementations, the not bits are developed merely by inverting the bits. Figure 16 illustrates this. The inverter is a power inverter because it will have to drive six different and circuits.



Development of a Not Bit

Figure 16

Development of Bits from the Digits

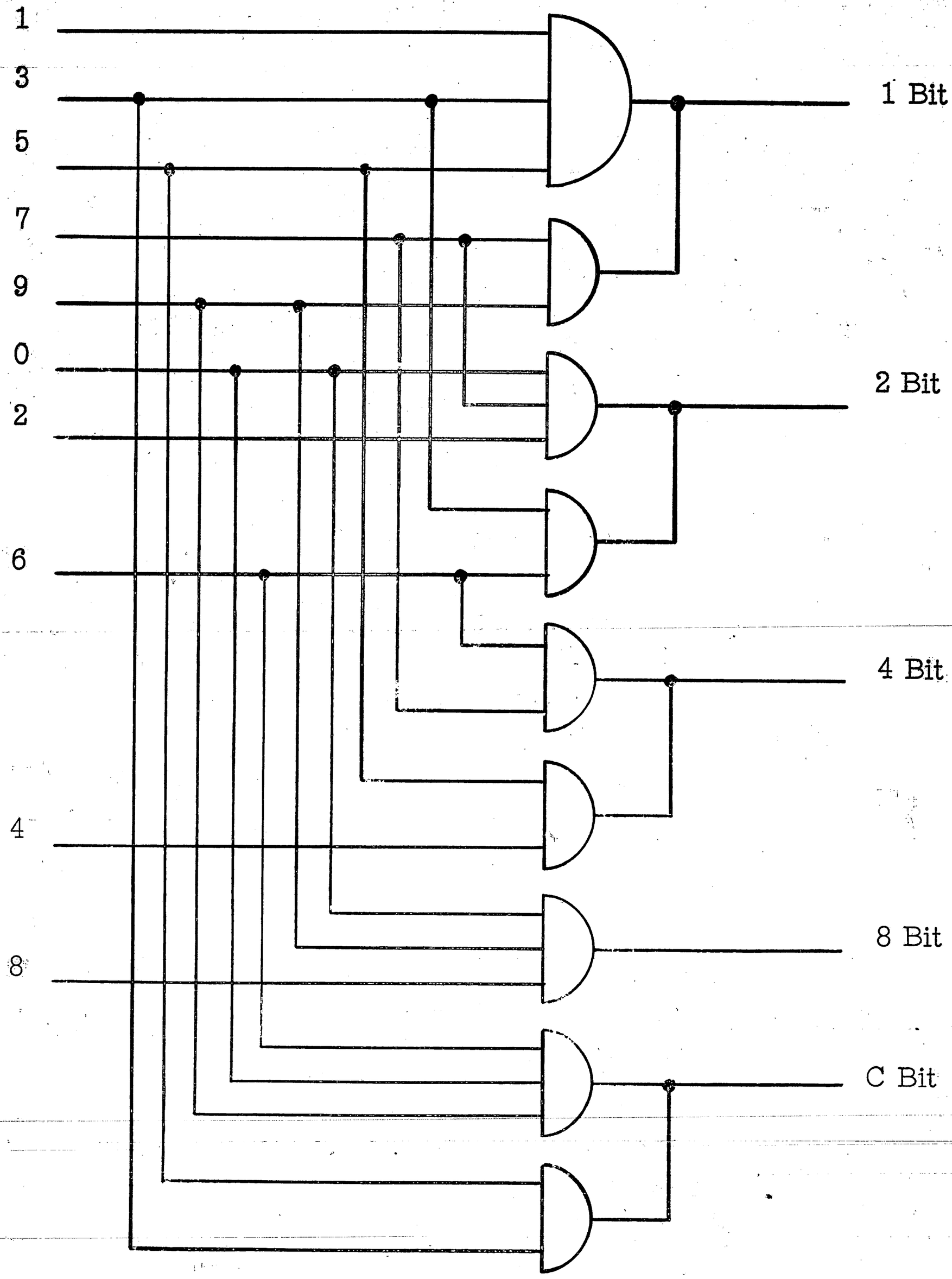


Figure 15

The second approach to the translator is the one which will be used because of its three major advantages. These advantages are:

1. Cost - Since it requires twelve less transistors than the first approach, it will be considerably cheaper to build.
2. Straightforwardness - Since the translation is so straightforward--two-out-of-five through one stage of logic to decimal digits, then decimal digits through one stage of logic to BCD bits; as compared to the two-out-of-five code through several stages of logic to BCD code--it will be easier to service in case of failure.
3. Reliability - With fewer components the chances of a component failure are lower than with a large number of components.

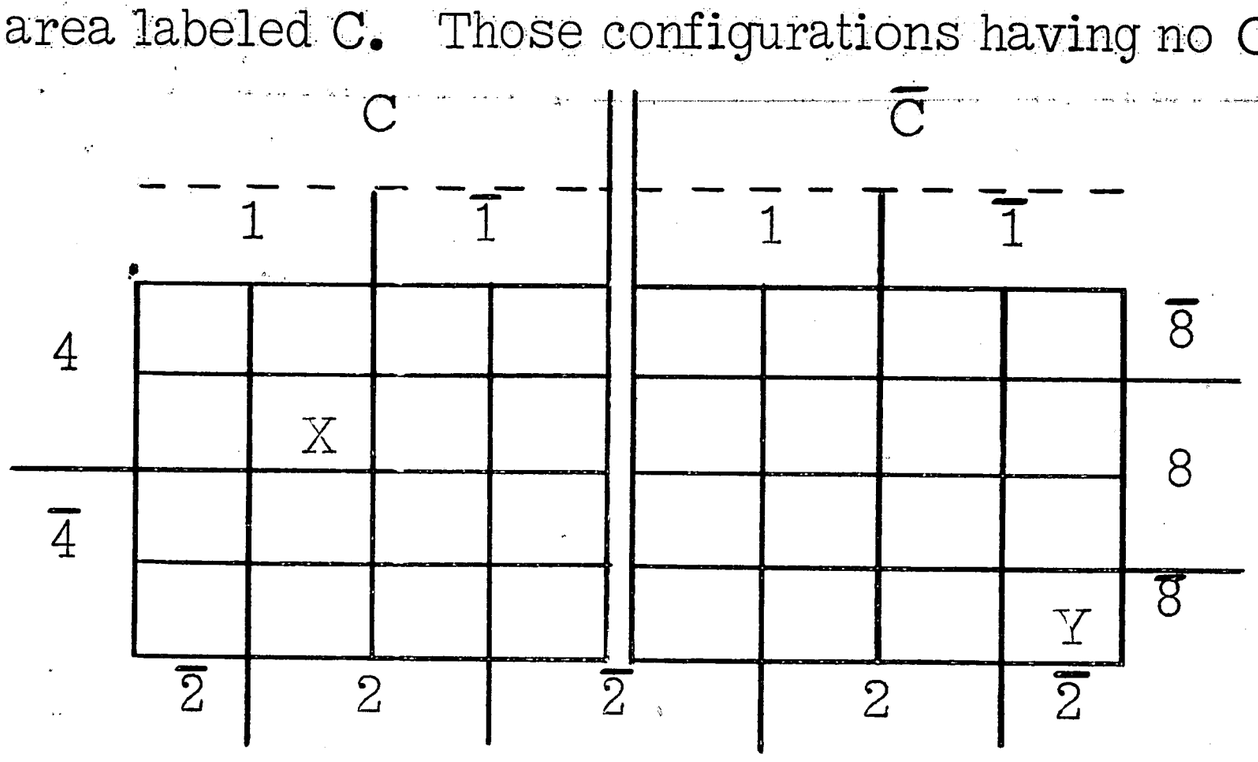
6 CHECKING THE TRANSLATOR

Part of the conditions set down for the translator have been met; specifically, the conversion of two-out-of-five bits to BCD bits. There is not yet any guarantee that the output of the translator is valid. Therefore, the next step in the design is to develop a network to go with the translator which will flag an error if an invalid output appears. There are thirty-two possible combinations of the five BCD bits. Ten of these combinations are valid according to the

requirements set down for the output of the translator. Twenty-two of the possible combinations are invalid. Table IV lists all the valid and invalid combinations. Any error detecting network that is designed must be able to discriminate between the valids and the invalids. Again, as with the translator, several design approaches will be tried in an effort to develop the best checking network.

6.1 MAP DEVELOPMENT OF THE CHECKING NETWORK

The first approach to be used is that of placing all the invalid combinations on a map[1] and then reducing these to a simple Boolean expression. The map which will be used is shown in Figure 17. Each character containing a C bit is placed to the left of the center line in the area labeled C. Those configurations having no C bits will



Map for Developing Boolean Expression for "Error"

Figure 17

[1] R. K. Richards, Arithmetic Operations in Digital Computers, (Princeton, New Jersey, 1958), pp. 66 - 70

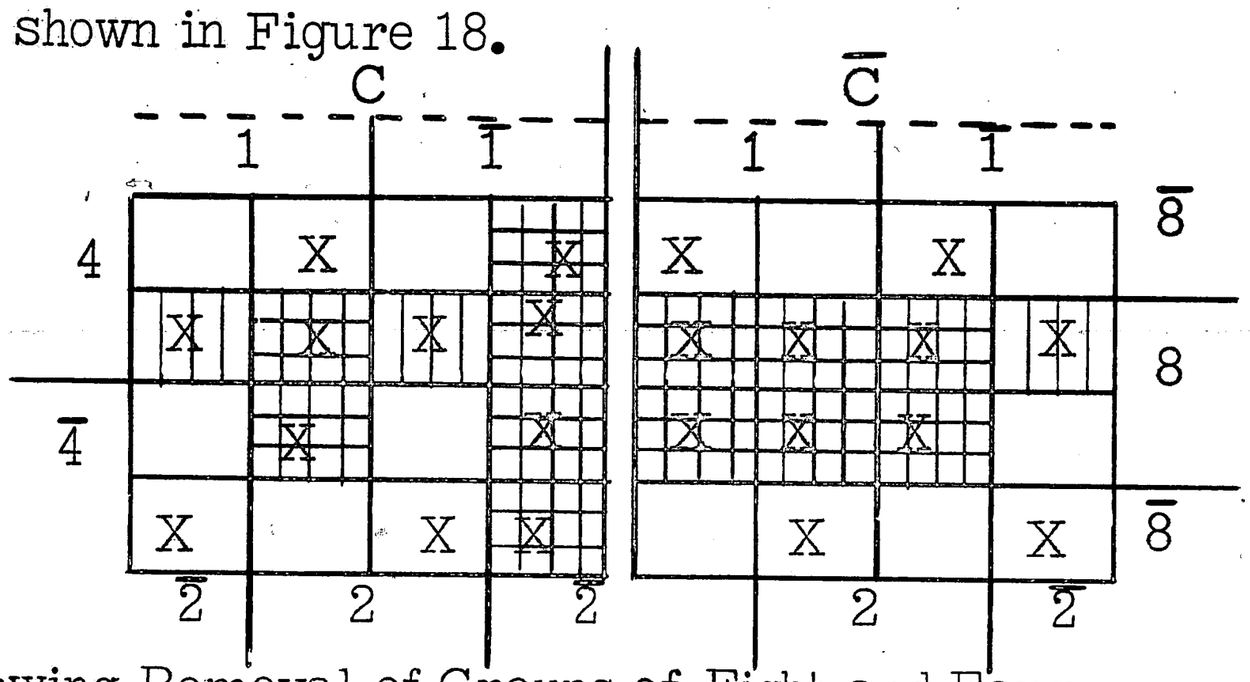
BCD COMBINATIONS

Valid	Invalid
$1\ \overline{2}\ \overline{4}\ \overline{8}\ \overline{C}$	$\overline{1}\ \overline{2}\ 4\ \overline{8}\ \overline{C}$
$\overline{1}\ 2\ \overline{4}\ \overline{8}\ \overline{C}$	$1\ 2\ \overline{4}\ \overline{8}\ \overline{C}$
$1\ 2\ \overline{4}\ \overline{8}\ C$	$1\ \overline{2}\ 4\ \overline{8}\ \overline{C}$
$\overline{1}\ \overline{2}\ 4\ \overline{8}\ \overline{C}$	$\overline{1}\ 2\ 4\ \overline{8}\ \overline{C}$
$1\ \overline{2}\ 4\ \overline{8}\ \overline{C}$	$1\ \overline{2}\ \overline{4}\ 8\ \overline{C}$
$\overline{1}\ 2\ 4\ \overline{8}\ C$	$\overline{1}\ 2\ \overline{4}\ 8\ \overline{C}$
$1\ 2\ 4\ \overline{8}\ \overline{C}$	$1\ 2\ \overline{4}\ 8\ \overline{C}$
$\overline{1}\ \overline{2}\ \overline{4}\ 8\ \overline{C}$	$\overline{1}\ \overline{2}\ 4\ 8\ \overline{C}$
$1\ \overline{2}\ \overline{4}\ 8\ C$	$1\ \overline{2}\ 4\ 8\ \overline{C}$
$\overline{1}\ 2\ \overline{4}\ 8\ C$	$\overline{1}\ 2\ 4\ 8\ \overline{C}$
$1\ 2\ 4\ 8\ \overline{C}$	$\overline{1}\ \overline{2}\ \overline{4}\ \overline{8}\ C$
$\overline{1}\ \overline{2}\ \overline{4}\ \overline{8}\ C$	$1\ \overline{2}\ \overline{4}\ \overline{8}\ C$
$1\ \overline{2}\ \overline{4}\ \overline{8}\ C$	$\overline{1}\ 2\ \overline{4}\ \overline{8}\ C$
$\overline{1}\ \overline{2}\ 4\ \overline{8}\ C$	$1\ 2\ 4\ \overline{8}\ C$
$1\ 2\ 4\ \overline{8}\ C$	$\overline{1}\ \overline{2}\ \overline{4}\ 8\ C$
$\overline{1}\ 2\ \overline{4}\ 8\ C$	$1\ 2\ \overline{4}\ 8\ C$
$1\ \overline{2}\ 4\ 8\ C$	$\overline{1}\ \overline{2}\ 4\ 8\ C$
$\overline{1}\ 2\ 4\ 8\ C$	$1\ \overline{2}\ 4\ 8\ C$
$1\ 2\ 4\ 8\ C$	$\overline{1}\ 2\ 4\ 8\ C$
	$1\ 2\ 4\ 8\ C$

Table IV

be placed in the sector to the right labeled $\overline{C}$. Each bit structure containing a one bit will be in the left half of each sector while those not having a one bit will be in the right half of each sector. Each bit configuration having a four bit will be in the upper half of the map and those not having a four bit will be in the lower half, etc. As an example, the X in the above map is located at the spot whose coordinates are C, 8, 4, 2, and 1. The Y is located at the position identified by $\overline{C}$, $\overline{8}$, $\overline{4}$, $\overline{2}$, and $\overline{1}$.

If all the invalid combinations are now plotted on the map, it will appear as shown in Figure 18.



Map for Showing Removal of Groups of Eight and Four

Figure 18

A simplified Boolean expression for these invalid combinations can be developed by removing groups of X's that conform to a pattern. The first group to try to remove would be a group of sixteen since this group could be represented by only one bit. There are no groups of that size on this map. An example of a group like this would be if all the positions on the left hand side of the map contained an X,

all the terms in the equation representing the invalid combinations could be removed and replaced with the single term C.

Since there are no groups of sixteen to remove, the next group to look for are groups of eight. This shaded portion of the map shows the only group of eight that may be removed. These eight terms may be replaced with the single term $8 \cdot 4$ since every square on the map that can be addressed by these two bits contains an X. So far eight of the twenty-two terms in the expression for an error have been replaced by a single term, $8 \cdot 4$.

The next group to be removed after the groups of eight have been removed are the groups of four. These groups can be represented by an expression containing only three bits. The cross-hatched areas on the map show these groups. There are four of these on the map. They are $\overline{C} \cdot 8 \cdot 2$, $8 \cdot \overline{2} \cdot 1$, $\overline{C} \cdot 8 \cdot 1$, and $C \cdot \overline{2} \cdot \overline{1}$. At this point the error expression is: Error = $8 \cdot 4 + \overline{C} \cdot 8 \cdot 2 + 8 \cdot \overline{2} \cdot 1 + \overline{C} \cdot 8 \cdot 1 + C \cdot \overline{2} \cdot \overline{1}$ + all remaining X's.

The groups to be removed after all the groups of four have been removed are the groups of two. These groups are represented by four bits. There are eight of these groups on the map. These groups are: $\overline{8} \cdot \overline{4} \cdot \overline{2} \cdot 1$, $\overline{C} \cdot 4 \cdot \overline{2} \cdot 1$, $C \cdot \overline{8} \cdot \overline{4} \cdot \overline{2}$, $C \cdot \overline{8} \cdot \overline{4} \cdot \overline{1}$, $C \cdot 4 \cdot \overline{2} \cdot 1$, $\overline{C} \cdot \overline{4} \cdot 2 \cdot 1$, and $\overline{C} \cdot 4 \cdot 2 \cdot \overline{1}$. The final expression then becomes:

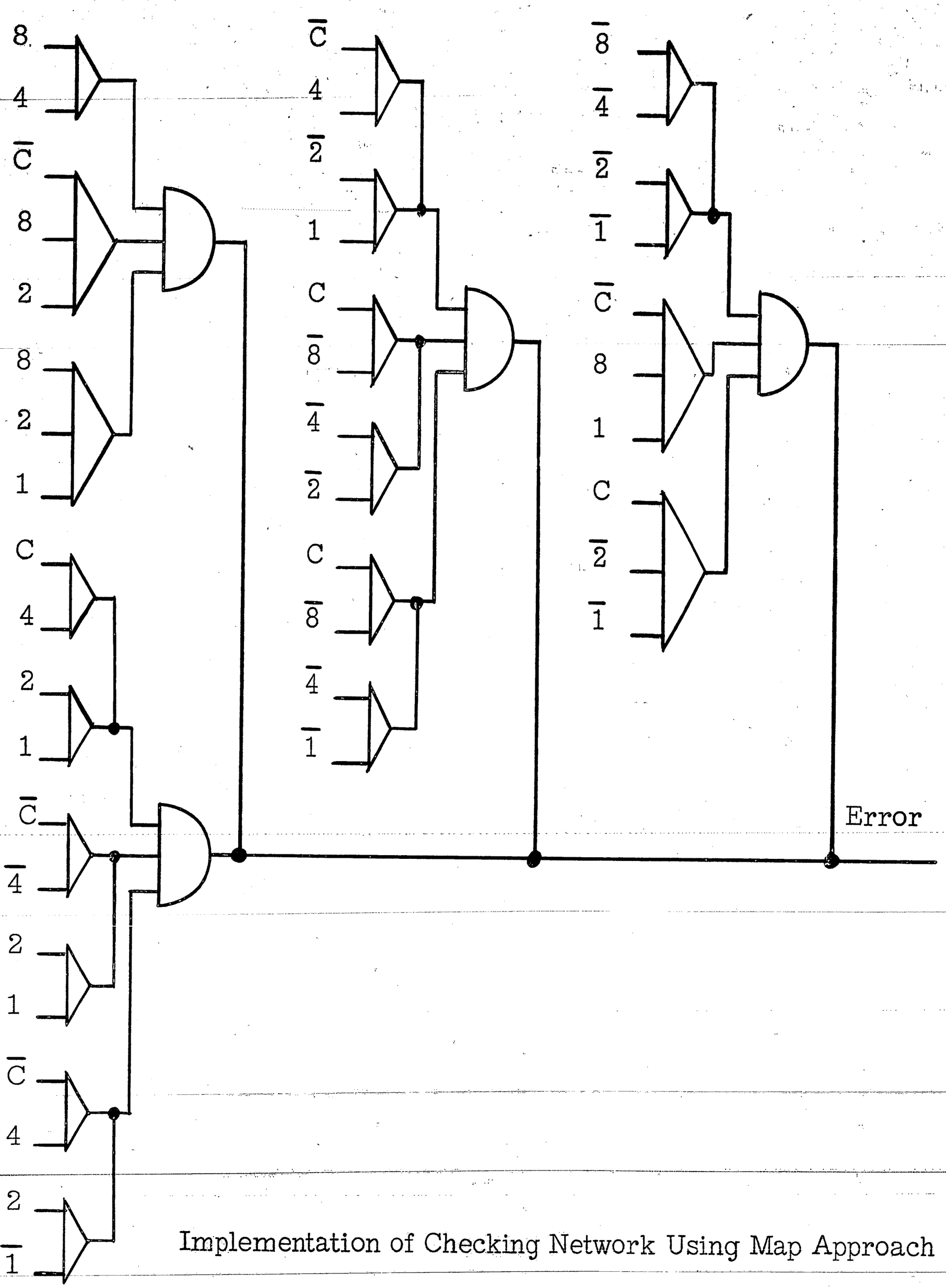
$$\text{Error} = 8 \cdot 4 + \overline{C} \cdot 8 \cdot 2 + 8 \cdot \overline{2} \cdot 1 + \overline{C} \cdot 8 \cdot 1 + C \cdot \overline{2} \cdot \overline{1} + \overline{8} \cdot \overline{4} \cdot \overline{2} \cdot \overline{1} + \overline{C} \cdot 4 \cdot \overline{2} \cdot 1 + C \cdot \overline{8} \cdot \overline{4} \cdot \overline{2} + C \cdot \overline{8} \cdot \overline{4} \cdot \overline{1} + C \cdot 4 \cdot 2 \cdot 1 + \overline{C} \cdot \overline{4} \cdot 2 \cdot 1 + \overline{C} \cdot 4 \cdot 2 \cdot \overline{1}$$

The implementation of this expression requires twenty-three transistors. It is shown in Figure 19. The not bits for this circuit are also obtained by inverting the bits.

6.2 DEVELOPMENT OF THE CHECKING NETWORK BY ANALYSIS OF INVALID COMBINATIONS

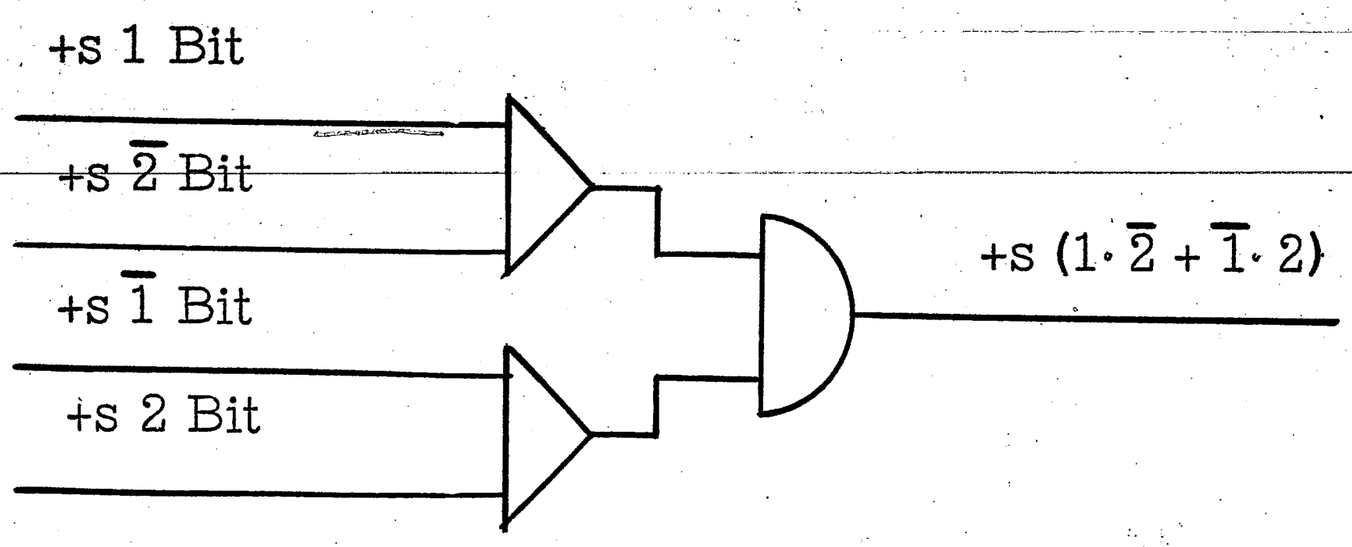
Since the error circuit previously developed turned out to be expensive and cumbersome, another approach will be taken. In this approach, invalid combinations will be analyzed to determine what their characteristics are and if a simple design can be achieved which will take advantage of these characteristics.

The first characteristic of these twenty-two invalid combinations is that sixteen of them have even parity; that is, there are an even number of bits in the combination. Since it was stated in the definition of the translator that odd parity BCD code would be used, then sixteen of the thirty-two combinations are automatically invalid. Therefore, a circuit which will flag an even parity combination when it comes upon one could be incorporated as part of the check. Such a circuit can be implemented by pyramiding "exclusive or" circuits. An "exclusive or" of the one bit and the two bit would give a positive output only if one or the other bit was present, but not if neither or both are present. Figure 20 illustrates this circuit. If an "exclusive or" is also built for the four and eight bits, an output will be obtained if only one of these two bits is present. Since an "exclusive or" output from either of these circuits



Implementation of Checking Network Using Map Approach

Figure 19



Exclusive "Or" of Two Bits

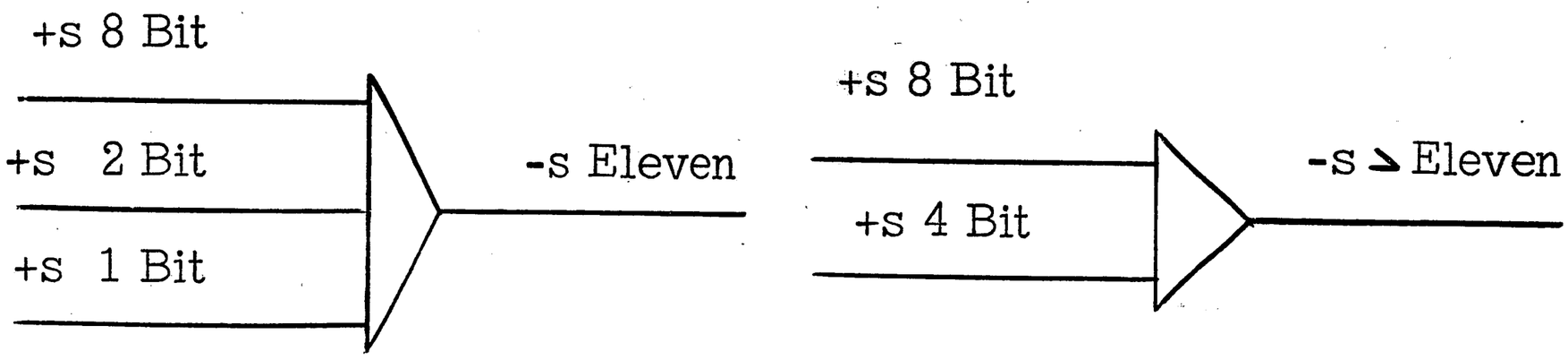
Figure 20

means only one of the two inputs to each is present, exclusive or-ing these outputs will give a line that will be positive if one or three bits are present. If this output is then exclusive or-ed with the C bit, the output of this circuit will be positive if one, three, or five bits are present. The output will be negative if two, four, or no bits are present. This line can then be labelled "-s Even Parity," meaning a negative signal is present whenever the parity is even. This line can now be used as one of the conditions to represent an error.

A second characteristic of the invalid combinations can be seen by examining the valid combinations. If the BCD bits for all the digits are summed, it can be seen that the zero gives the highest total, having the eight plus the two equal ten. Therefore, sums of eleven through fifteen are invalid. The following are the bits present for each of these sums:

eleven	8 2 1
twelve	8 4
thirteen	8 4 1
fourteen	8 4 2
fifteen	8 4 2 1

From this little table, it can be seen that any combinations of BCD bits containing 8 • 2 • 1 or 8 • 4 are invalid. These two expressions can also be incorporated in the check network.

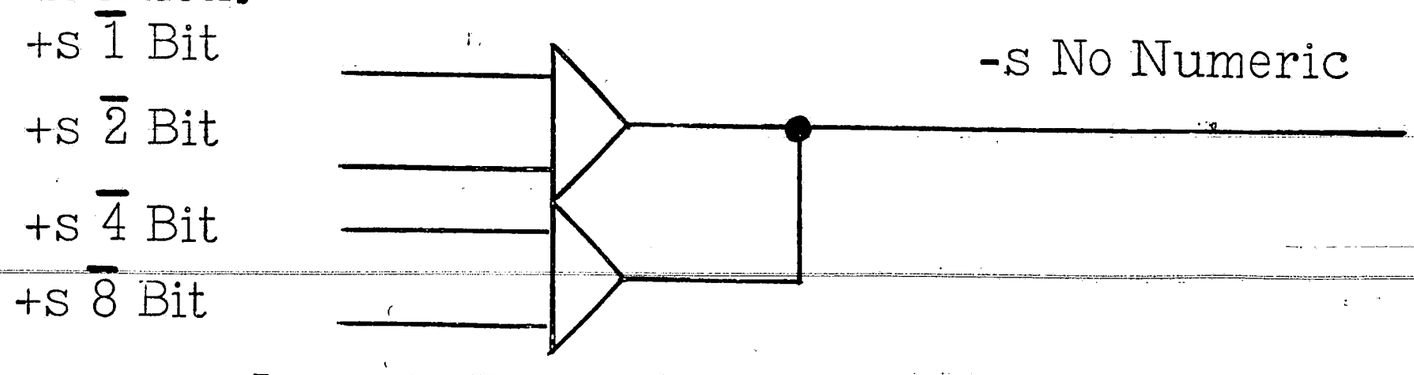


Implementation of Detection of Digits greater than Ten

Figure 21

This characteristic covers four more invalid terms not covered by the first characteristic.

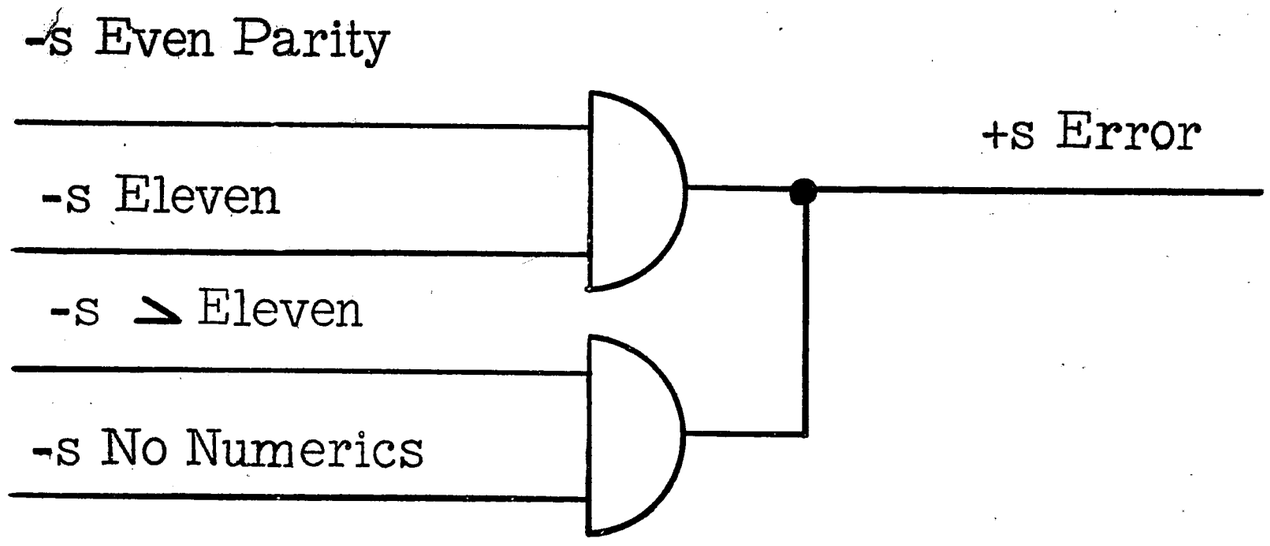
The only remaining invalid combination is C• $\overline{8}$• $\overline{4}$• $\overline{2}$• $\overline{1}$. An and circuit which will recognize no numeric bits will then catch this combination.



Logic to Detect Absence of Numeric Bits

Figure 22

By combining the circuits which have been developed to search out the three characteristics which completely define all the invalid combinations, the last stage of the error circuit would be as shown in Figure 23. The complete network which is shown in Figure 24 includes twenty transistors, a savings of three over the first



Last Stage of Checking Network

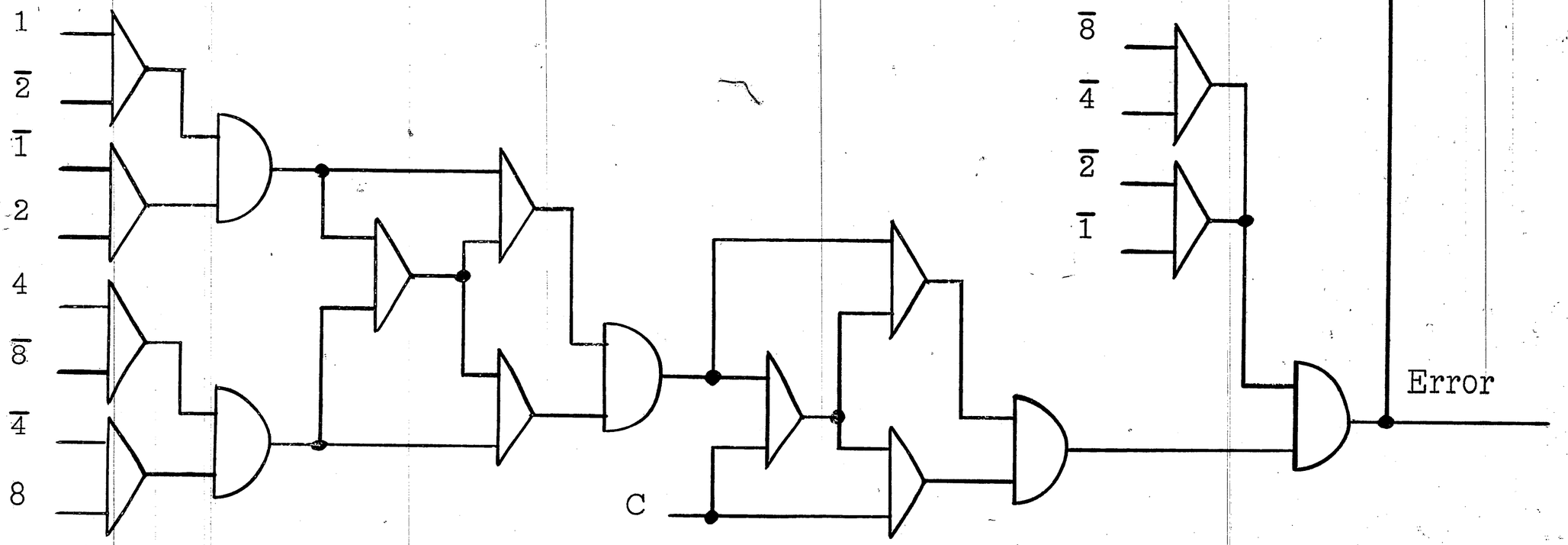
Figure 23

approach. Although this is a small savings, it would be a more straightforward and logical network to troubleshoot in a case of a failure. The expression for this network is as follows:

$$\text{Error} = \text{Even Parity} + 8 \cdot 2 \cdot 1 + 8 \cdot 4 + \overline{8} \cdot \overline{4} \cdot \overline{2} \cdot \overline{1}$$

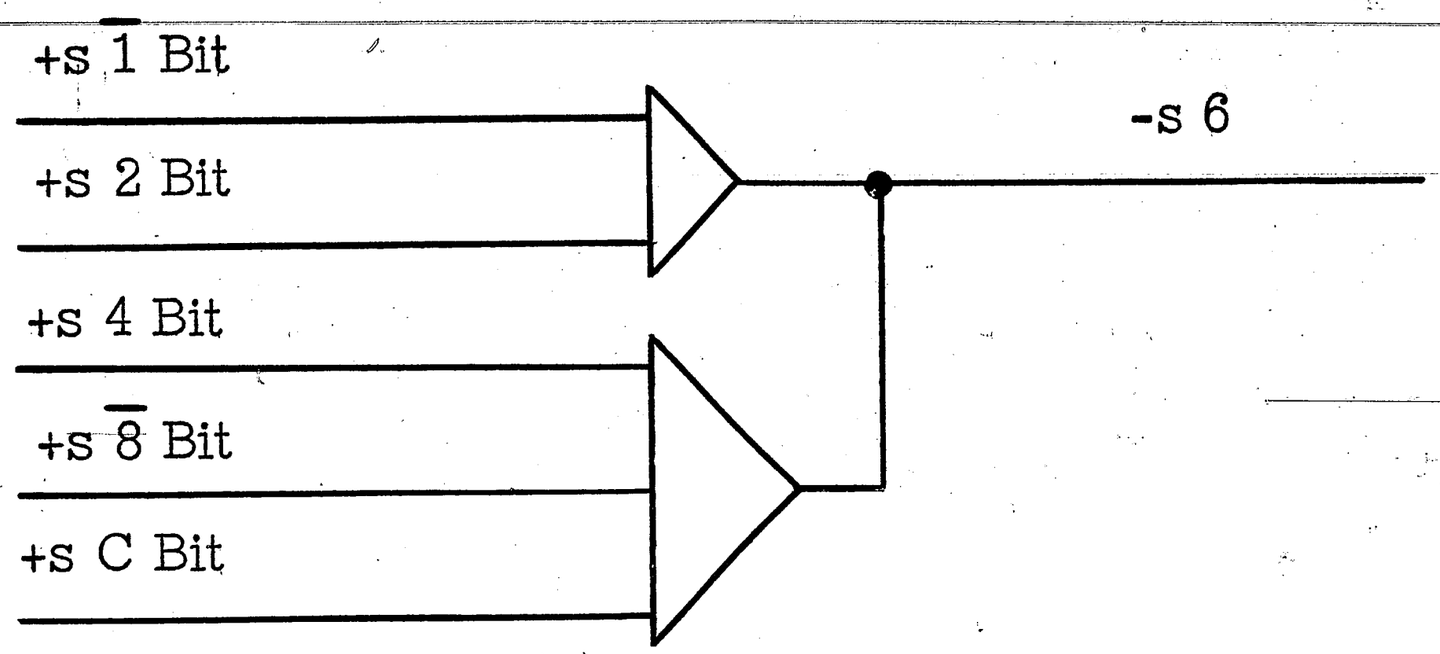
6.3 DEVELOPMENT OF THE CHECKING NETWORK USING THE INVERSION OF "VALID"

A third approach that can be taken is to use the reverse philosophy; that is, look for the valid condition and flag an error if it does not appear. This can be accomplished by developing each of the ten valid decimal digits by using their BCD bits as inputs. An example is shown for the digit six (Figure 25).



Implementation of Checking Network Using Analysis of Invalid Characters

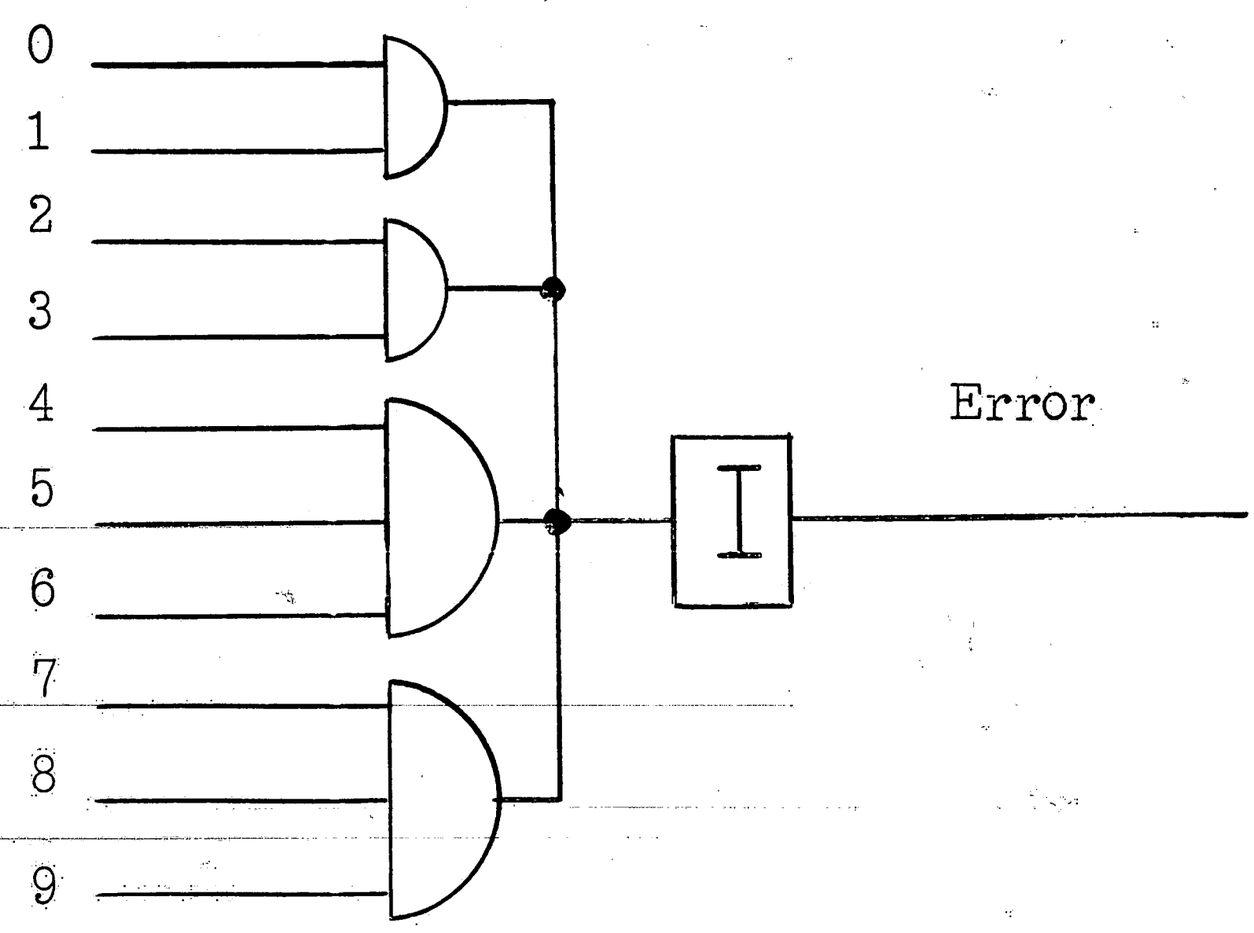
Figure 24



Implementation of the Digit Six from BCD Code

Figure 25

If all ten digits are or-ed together, an inverter added to the output of the or circuit will give the desired error signal.



Error Circuit Implemented by Invertering "Valid" Signal

Figure 26

This check circuit also requires twenty-five transistors, twenty to develop the digits and five to detect the errors. The only advantage of the previous circuit over this circuit is that it takes a more direct approach to its job; that is, it looks for the error rather than the not error.

7 ANALYSIS OF THE DESIGN

The two networks which make up the translator are now complete. They are the translator itself and its check network that appears on its output. The next work to be done now is to carefully study and examine these circuits to see what problems or possible undetected errors occur when a single component fails or when it is presented with an invalid input.

The criteria that was established earlier for the translator states four specific instances in which an error must be flagged. These instances will now be reviewed one at a time to insure that the design is complete. They will be reviewed first without taking circuit failures into account, and then with the circuit failures included in the analysis.

7.1 ANALYSIS IGNORING CIRCUIT FAILURES

The following listing shows the results of analysis without regarding circuit failures.

1. An error must be flagged if the input is presented with an invalid two-out-of-five code combination.

   This condition is met since any invalid two-out-of-five

input code fails to select a digit because the digits require three not bits as well as the two bits to satisfy the five-way and. If no digits are selected, there will be no output bits. This situation would be detected by the even parity check and the $\overline{8}\cdot\overline{4}\cdot\overline{2}\cdot\overline{1}$ check in the error circuit.

2. An error must be flagged if the output has even parity.

This situation is detected by the even parity check circuit in the error detecting network on the translator output.

3. An error must be flagged if the output character is a digit other than zero through nine.

This situation is checked by the circuits which look for code combinations with an eight and a four bit, combinations with an eight bit, a two bit, and a one bit, or combinations with no numeric bits.

4. An error must be flagged if the output character is different from the input character.

This is checked by the way in which the translation occurs; that is, the input is reduced from two-out-of-five code to a digit. This digit then causes a BCD output to represent itself. By using the five-way ands to develop the digits, a digit can be developed only if it is completely defined by a valid two-out-of-five bit configuration.

7.2 ANALYSIS OF CIRCUIT FAILURES

On the basis of the first review, the translator satisfies all requirements. But the established criteria states that in addition to the conditions defining the input and output, any single circuit failure must either not effect the translation or cause an error to be flagged. The criteria are, therefore, reviewed again taking into account circuit failures. The results are as follows:

1. An error must be flagged if the input is presented with an invalid two-out-of-five code combination.

   A circuit failure occurring in three out of the five stages of logic required for checking allows an undetected error. These failures are as listed.

   a. Not bit generation - If three out of five bits is presented to the translator and the inverter generating the not bit of one of the three bits presented has a shorted transistor, the output will not fall and one of the digits will be satisfied. For example, if a zero bit, a one bit, and a two bit are presented and the not two bit line stays positive, a one will be translated and no error will be detected. The input may have been either a zero, one, or two that had picked up an extra bit.

b. Digit generation - If an invalid combination does not select a digit, but in a digit generator one of the and circuits has an open transistor, the digit may be generated and an error not flagged. An example in this case could be if only a three bit is presented and the two-way and in the digit generator for the digit nine has an open transistor, the not bits will cause the three-way block to turn off and the digit line will be satisfied with a minus level.

c. BCD bit information - Again an invalid bit combination would not cause any of the digits to be generated, unless the or circuit on any one of the numeric bits (1, 2, 4, or 8) has a shorted transistor. This will put a single bit at the output of the translator and no error will be detected since a single numeric bit is a valid character.

d. Error detection - If no digit is generated because of an invalid input, a failure in the even parity check would not cause an undetected error because the $\overline{8}\cdot\overline{4}\cdot\overline{2}\cdot\overline{1}$ and circuit would detect this situation and visa versa.

If the $\overline{8}\cdot\overline{4}\cdot\overline{2}\cdot\overline{1}$ circuit failed, the even parity network check would not catch $C\cdot\overline{8}\cdot\overline{4}\cdot\overline{2}\cdot\overline{1}$, but this combination can be generated only by a failure in the translator. This would then be a double for which no protection

is provided. The criteria requires no double error protection. However, if one of the or circuits had an open, the other one would conduct and the error would be caught. Therefore, a failure in the error circuit will not cause an undetected error.

2. An error must be flagged if the output character has even parity. This situation is handled adequately even when considering circuit failures because the only way that even parity can appear at the output without a circuit failure in the translator is if the input is invalid and no bits have been generated. It has just been shown that the error circuit will handle this situation. To fail to detect this situation would require two simultaneous circuit failures. The criteria set up does not require detection of two circuit failures.

3. An error must be flagged if the output character is other than a digit zero through nine.

This situation cannot exist without a failure in the translator since the digit generators require the failure of two lines before more than one digit is generated. The only way to have a character different than or greater than zero through nine is for two digits to be simultaneously generated or an or at the bit generation end to fail. These errors will be detected by the appropriate checks in the error network.

4. An error must be flagged if the output character is different from the input character.

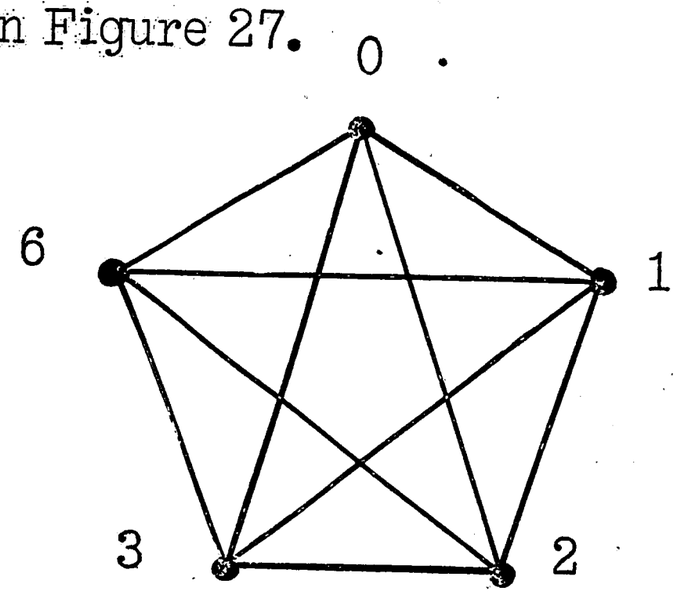
This situation is adequately handled since it can occur only due to a failure in the translator as described in the above paragraph.

8 NEED FOR AN INPUT CHECKING NETWORK

As can be seen from the above analysis, the translator with its check network meets all the error criteria when the input is valid. If the input were guaranteed to be valid at all times, the combination of the translator and its check network would truly be fail safe even if each of its components by themselves (the translator and the check network) are not. A valid input can be guaranteed if a check is placed ahead of the translator to detect invalid two-out-of-five code combinations. This check would have to be able to detect a single transistor failure within itself also. It does not appear at first that this is a necessary restriction. Since, if the whole combination of the translator and two check networks is fail safe, then if a transistor fails in the two-out-of-five check, it would take a simultaneous failure in the translator to allow an undetected error. This is out of the bounds of the criteria set up. However, if a transistor failure occurs in the two-out-of-five check which will cause the network to never signal an error, this failure may go unnoticed for a long period of time since errors are not flagged frequently in most computers. Therefore, if a failure occurs later in the translator, an undetected error can occur.

This two-out-of-five check network should then be able to detect a failure within itself.

There is a standard two-out-of-five check network[2] that is frequently used in applications where the two-out-of-five code is used. This fail safe network is developed by drawing the five bits as equally spaced on the circumference of a circle. Using these five points of a star, and also as corners of a pentagon, the two figures are drawn as shown in Figure 27.



Lines Representing "And" Circuits in the Two-Out-of-Five Check

Figure 27

If each line of each figure is represented as an and circuit of the bits at each end of the line, there will be ten and circuits. The five ands that are sides of the pentagon are or-ed together and the five ands that are lines in the star are or-ed together. If two out of the five bits are present, one of the and circuits will be energized since two bits define a line either

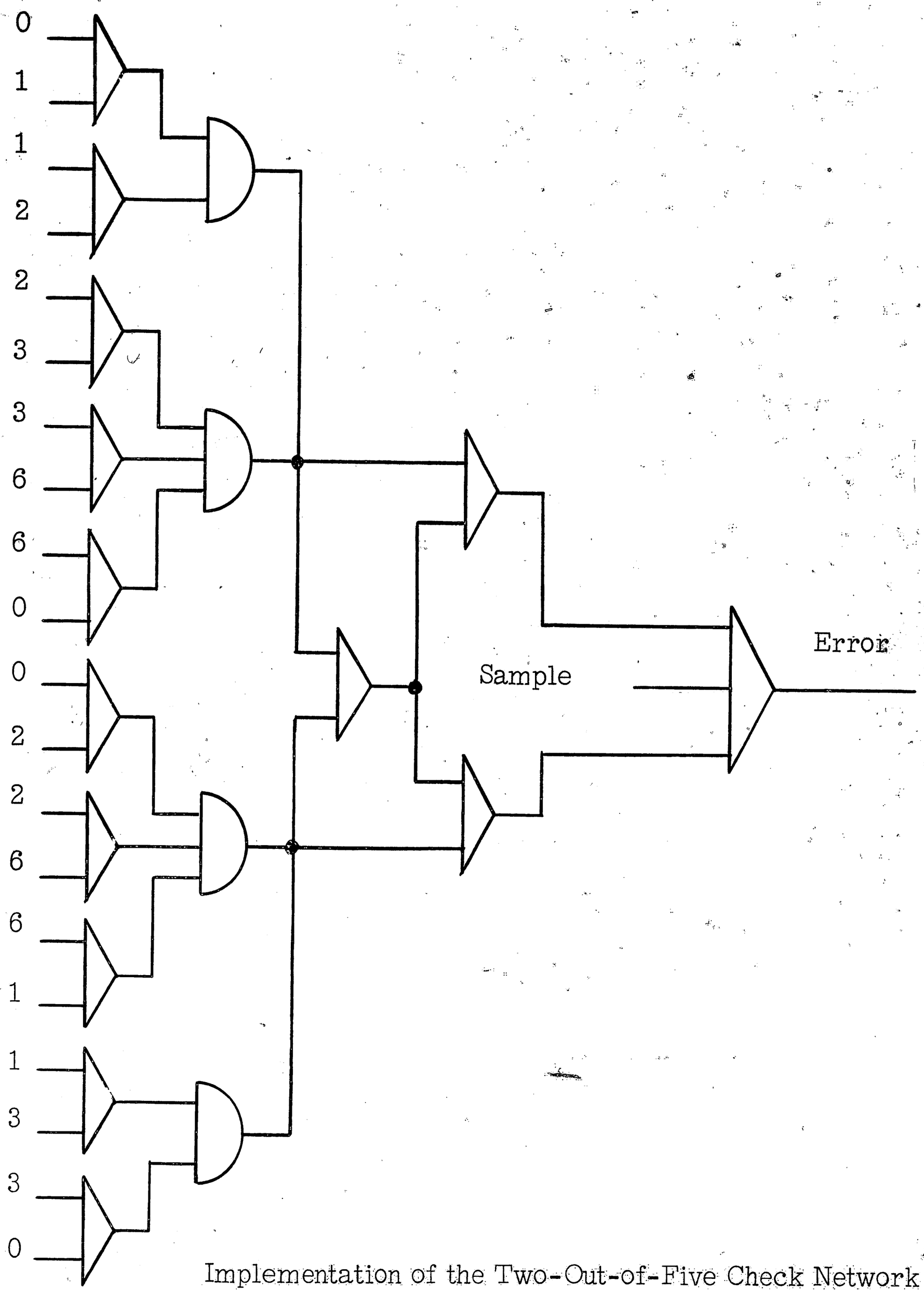
---

[2]George J. Saxenmeyer, "Computer Principles" (Endicott, New York, 1959 -- Notes from course of the same title)

on the star or on the pentagon. If only one bit is present, none of the and circuits are energized because no line is defined. If more than two bits are present, more than one and circuit will be energized defining lines in both figures. A circuit that looks for an output of both or circuits or neither will then detect an error. This check network is shown in Figure 28.

The two-out-of-five check network is fail safe because it required two lines to make its decision. Since the digits zero, one, five, six, and nine are on the pentagon and the digits two, three, four, seven, and eight are on the star, the digits passing through a system will fall on each figure about half of the time. Therefore, if either side of the circuit has a failure holding one of the or circuits positive or negative all this time, this failure should be detected within the next few digits when there will not be an exclusive or condition of the two circuits. This check will then detect a failure within itself. Some means would have to be provided to manually check the last four transistors since from this point back only one line is being developed, and there is nothing with which to compare it. This can be done by using switches to force error conditions. These switches could be actuated periodically (once a day or so) to check the output of the error circuit.

The total network now contains 82 transistors -- 34 for translation and 48 for checking purposes. The whole network is fail safe, but each individual component is not. The translator itself is not fail safe. If further development of the translator itself could produce a translator



Implementation of the Two-Out-of-Five Check Network

Figure 28

only which would not allow an undetected error, the checking which costs more than the translation could be eliminated. It would also allow the translator to be attached to a data channel shared by many other networks and there would need to be a check only for the data on the channel and not for each unit attached to it. This is the situation that occurs in most computers; that is, a few registers, an adder, and a few translators may have their outputs all tied together on one master data channel.

9 FAIL SAFE DESIGN OF THE TRANSLATOR

The two check networks can be studied for a clue as to what is necessary for a fail safe design. It was pointed out in the discussion of the two-out-of-five check that two lines were developed and compared with each other. Also, the analysis of the output check showed that if the $\overline{8}\cdot\overline{4}\cdot\overline{2}\cdot\overline{1}$ circuit failed to detect no output bits, the even parity circuit would catch it and vise versa. It can also be seen that these lines are generated independently. Therefore, if two independently generated lines can be carried through the translator so that they can be compared, fail-safe translation could be achieved since a single component failure would cause a detectable difference between the two lines. A logical choice for the two lines in the translator would be the output and its not function. The output of the translator would then have ten lines--five BCD bits and their corresponding, independently generated not bits.

There is a postulate in Boolean Algebra that states that a function and its negative cannot occur or exist simultaneously. Therefore, if the previously developed translator is enlarged to give also its BCD not bits as an output, an error can be detected by merely comparing the bit and not bit. If either or neither is present at a given time, the output is in error.

10 DEVELOPMENT OF THE NOT BIT TRANSLATOR

Generating the not bits by inverting the bits would not satisfy the single failure detecting criteria since the not bits would always be opposite the bits unless the inverter failed. If a failure, for instance, caused the "one bit" line only to be up, the "not one bit" would fall and no error would be detected because the not bit was independently generated.

Since a bit and not bit cannot exist simultaneously, the equations for the not bits can be written by including each digit in the not bit expression that does not appear in the bit expression. The following equations for the BCD not bits can then be written in terms of the digits:

$$\overline{1\ \text{Bit}} = 0 + 2 + 4 + 6 + 8$$

$$\overline{2\ \text{Bit}} = 1 + 4 + 5 + 8 + 9$$

$$\overline{4\ \text{Bit}} = 0 + 1 + 2 + 3 + 8 + 9$$

$$\overline{8\ \text{Bit}} = 1 + 2 + 3 + 4 + 5 + 6 + 7$$

$$\overline{C\ \text{Bit}} = 1 + 2 + 4 + 7 + 8$$

Implementing these expressions requires a five-way or for the "not one" bit, a five-way or for the "not two" bit, a six-way or for the "not four" bit, a seven-way or (two two-ways and a three-way collector or-ed) for the "not eight" bit, and a five-way or for the "not C" bit. Figure 29 shows the implementation of these equations.

The next point to be considered is whether or not the same digit generators that are used to develop the bits can be used to develop the not bits. A review of the analysis of the first translator shows that if a transistor opens in one of the digit generators, a digit may be generated by an invalid code combination that would satisfy the other half of the and circuit. For example, if three bits were presented to the input--a two bit, a three bit, and a six bit, if the three-way and circuit in the five-way and that develops the digit five has an open transistor, the two bit and the three bit will satisfy the two-way and. Its transistor will cut off and the other circuit will not be able to maintain the positive output. This line being negative will cause a one bit, a four bit, and a C bit to appear on the output. It will also cause a "not two" bit and a "not eight" bit to appear on the output. This has all the characteristics of a valid character--bit and not bit lines opposite each other, odd parity, valid digit between zero and nine. Therefore, an undetected error has occurred. The principle of generating the not bits independently has been violated.

Development of the Not Bits from the Digits

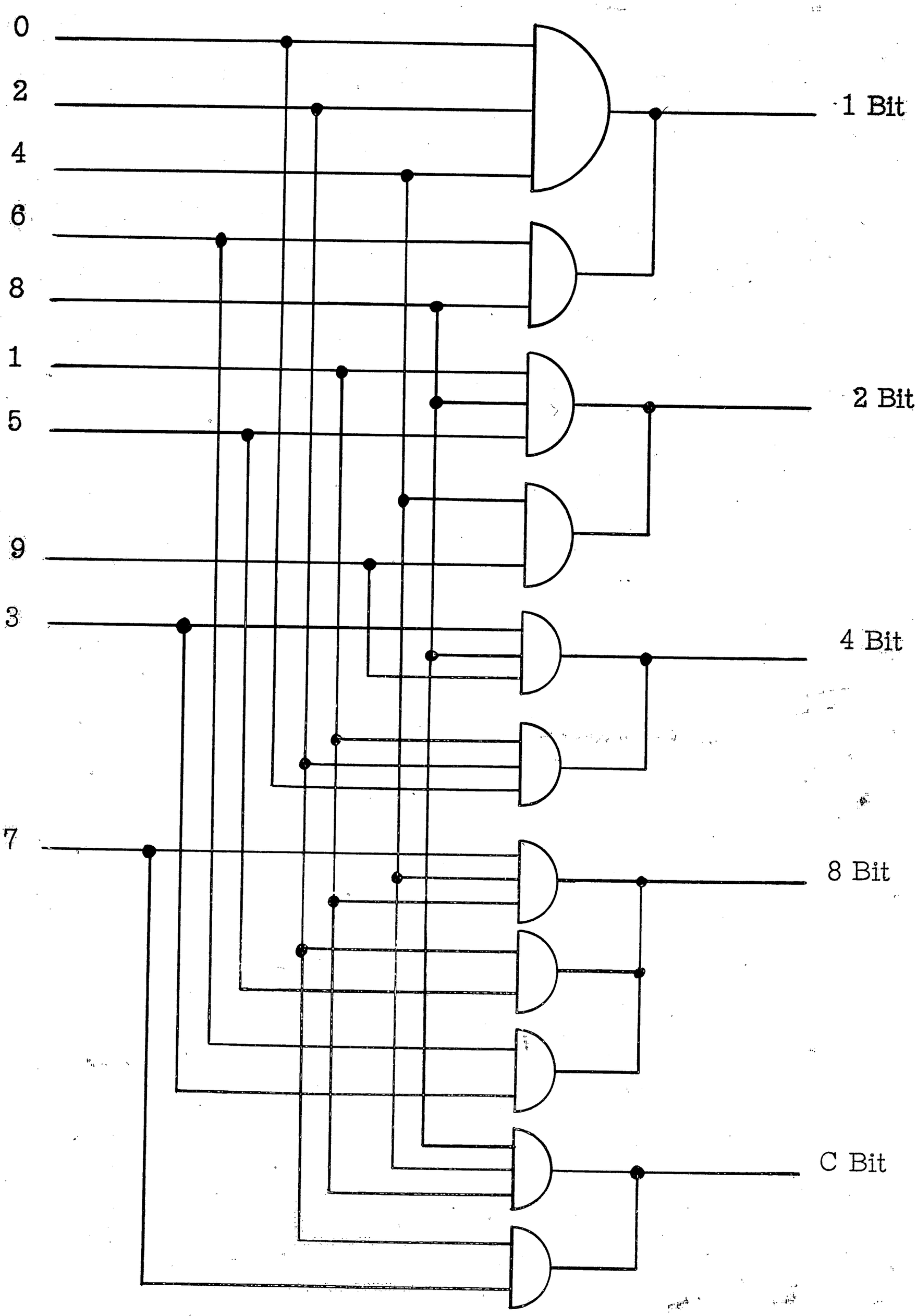


Figure 29

If separate digit generators are used, the error described in the previous example could not occur. If the digit generator for the bits had failed, no digit generator for the not bits would have appeared at the output. This produces the detectable situation of neither the bit nor the not bit appearing. The same would hold true if the not bit digit generator had failed. Therefore, the separate digit generators are necessary for fail-safe operation.

The last item to be considered in the fail-safe design is to determine if a separate inverter is needed to give the not function of the input bit. In the previous review of this point in the first translator, a shorted transistor in the inverter coupled with an invalid code of three digits would cause a digit to be generated. If a single inverter is used, the digit will be generated for both the bit translator and the not bit translator. An undetected error will occur. Therefore, it is also necessary that a separate inverter be used to develop the "not two-out-of-five" bit for each set of digit generators.

11 ANALYSIS OF THE FAIL SAFE TRANSLATOR

At this point a review of the two-channel translator will be made to determine if it meets the criteria set up for it.

1. An error must be flagged if the translator is presented with an invalid input.

If there are no circuit failures, an invalid input will cause no output lines to be generated. This results in the absence of both the bits and not bits. If any one transistor fails in the not bit generation, digit generation, or BCD bit generation of the bit translator to cause bits to appear at the output, the maximum number of bits that could be brought up would be three. An inverter failure or a digit failure can cause one or three bits to come up because no digit between zero and nine required more than three bits to represent it in the BCD code. An or circuit failure would cause only one bit to come up. No not bits would be brought up. The result would be the absence of at least two bits and their corresponding not bits. The same situation results if the failure is in the not bit generator. Therefore, under any circumstance, as long as there is only one transistor failure in the translator, an invalid input combination will always result in the absence of at least two bits and their respective not bits.

2. An error must be flagged if an even parity code combination appears on the output.

Because of the manner in which the translator decodes to a digit and then encodes to BCD bits, even parity can appear at the output only as a result of a circuit failure, either in the digit generators or in the bit generators. This failure will occur

in either the bit generator or the not bit generator, but not in both. Therefore, if the even parity is a result of having picked up an extra bit or an extra not bit, the situation will be such that both a bit and its corresponding not bit will be present. If the even parity is a result of dropping either a bit or a not bit, the result will be the absence of a bit and its corresponding not bit. As a result, even parity on the translator output will be accomplished either by the presence of a bit and its corresponding not bit or the absence of the same.

3. An error must be flagged if the output character is other than a digit from zero through nine.

The way in which the translator is developed allows only the input digit to appear at the output unless a failure has occurred in the circuitry. There is only one way for a digit to appear on the output which was not at the input. This will be caused by the failure of an or circuit which will result in an even parity output. If the transistor shorts, an extra bit (or not bit) will be picked up. If the transistor opens, the output will be missing a bit. In either situation, the output will have a mismatch between a bit and its not bit as described in the even parity error detection.

4. An error must be flagged if the output character is different from the input character.

This situation results in the same conditions as the previously described even parity detection.

12 CONCLUSIONS

The review of the translator providing an output of bits and not bits shows that any of the error conditions specified in the criteria will result in the absence of both a bit and its corresponding not bit or the presence of both a bit and its not bit. The translator itself is now fail safe in that it is impossible to get valid information through it when there is an error condition. This translator can now be attached to any BCD data channel, either numeric or alpha-numeric, that has a check circuit which will check the bits against the not bits. A fail safe check circuit must be provided on this channel to provide a true fail-safe system operation. The translator will not cause an undetected error by converting an invalid input to a valid output; therefore, the failure to detect an invalid output would result in undetected errors on this channel. These errors would be detected on another channel if all components which use this channel as an input are fail safe. Since it is desirable to detect errors where they occur, a fail-safe check circuit should be on this channel. This is not a serious restriction since if the data channel requires a fail-safe translator, it will carry both the bits and the not bits. An alpha-numeric channel is the same as the numeric channel except that it carries two extra bits (A and B)

which are used to prepresent alphabetic characters (Example: $A = \overline{C}\,B\,A\,\overline{8}\,\overline{4}\,\overline{2}\,1$, $Q = C\,B\,\overline{A}\,8\,\overline{4}\,\overline{2}\,\overline{1}$). The final two-channel translator is achieved at a cost of sixty-nine transistors.

Fail safe translation is achieved by two essentially duplicate translators--one generating bits and the other generating not bits. Two identical translators could be used and the outputs compared to see if they are the same. Also the outputs of one translator could be complemented and a bit-not bit compare made. This would require five extra transistors for the complements. The translator giving both bits and not bits has the advantage of putting both bits and not bits on the channel, eliminating the need for inversion every time the not bit is required in other circuits using the channel as an input.

A translator like the one developed here may be used not only within a machine, but also when transferring information from one machine to another, each using a different code. In an in-line processing system where information is constantly being updated with each transaction, it is vitally important that no mistakes (undetected) be made in the updating of the information because there is no way of repeating the transaction to see if the result is the same.

Translation to BCD Bits

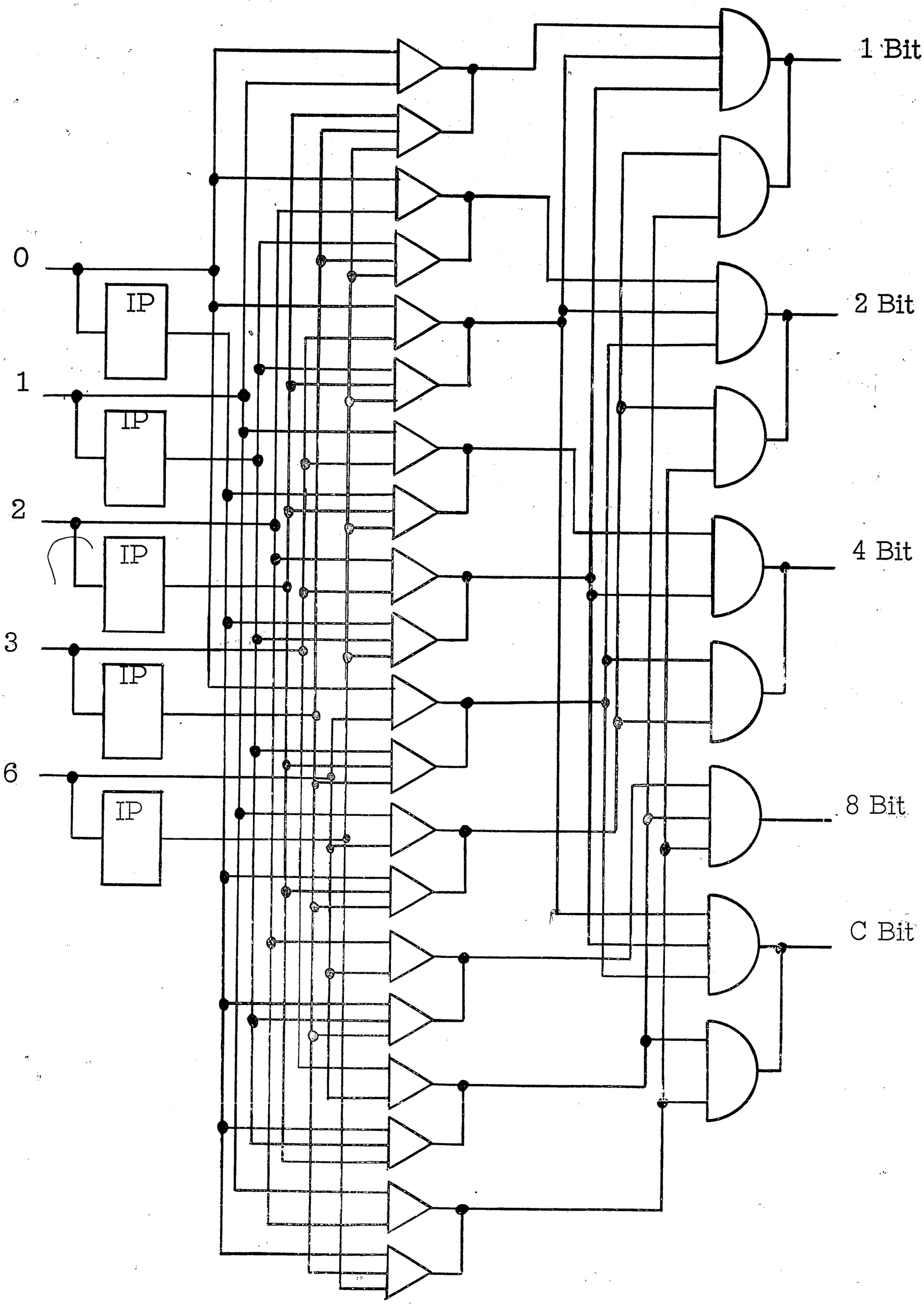


Figure 30

Translation to BCD Not Bits

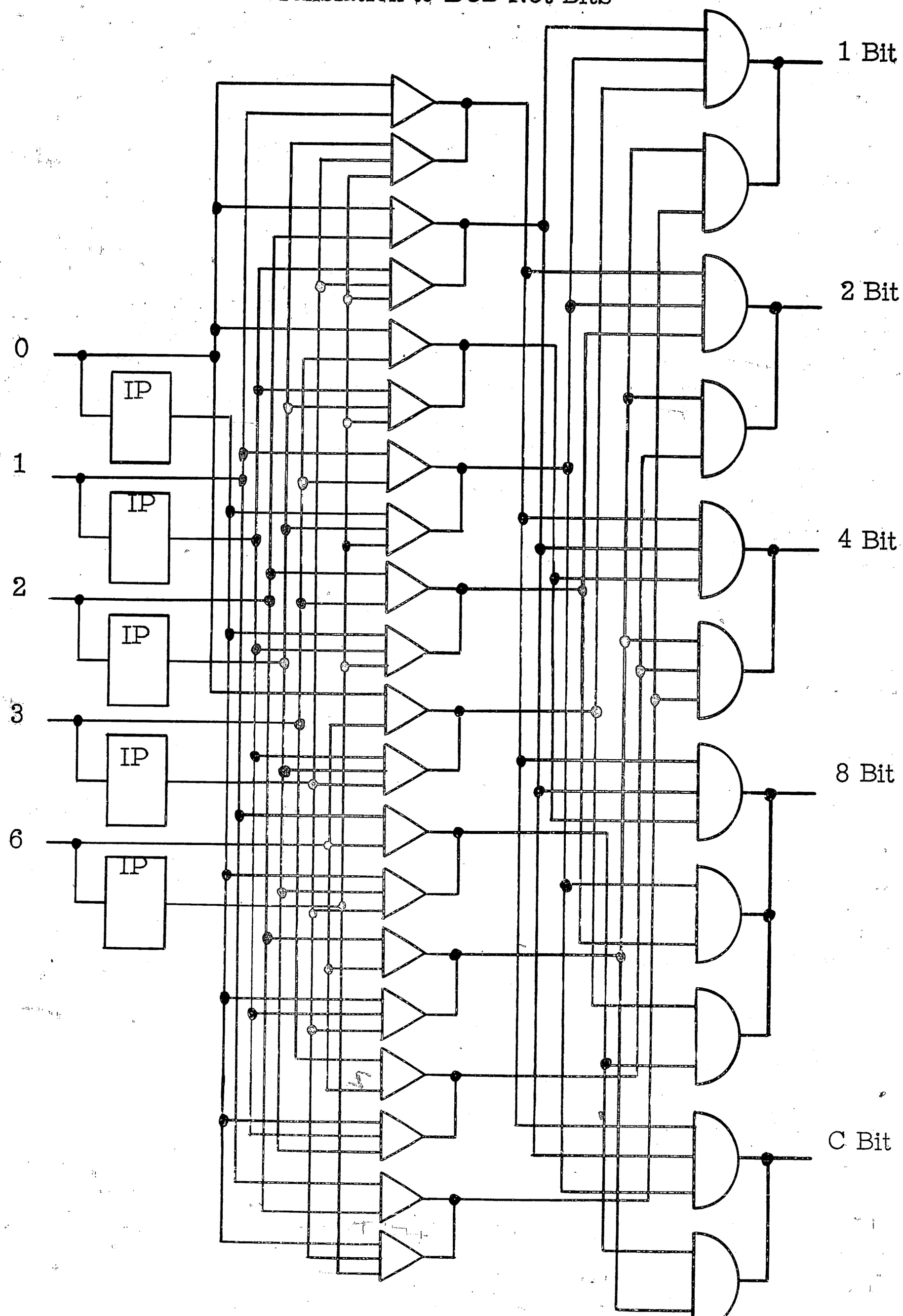


Figure 31

APPENDIX

TEST RESULTS

A test robot was built and studied to study the behavior of the translator and verify its performance characteristics. This robot consisted of the following parts.

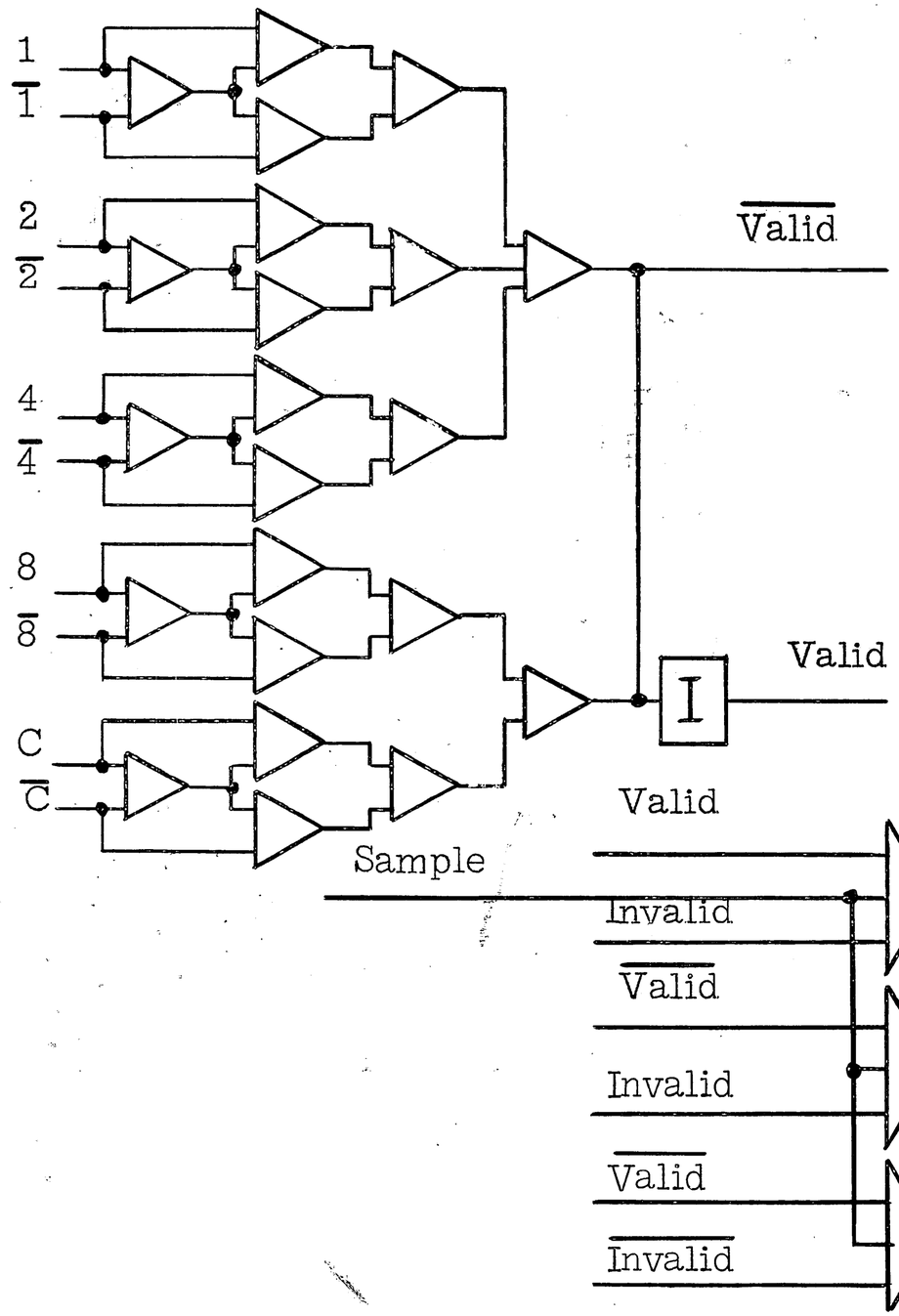
1. The two-channel fail safe translator designed in this paper.
2. A check network that could be used on a fail safe data channel. This network checks the translator output.
3. A six-position ring to serialize the input and output data which is parallel. Serializing this information provides a clear picture of the translation.
4. Five toggle switches to present input data to the translator.

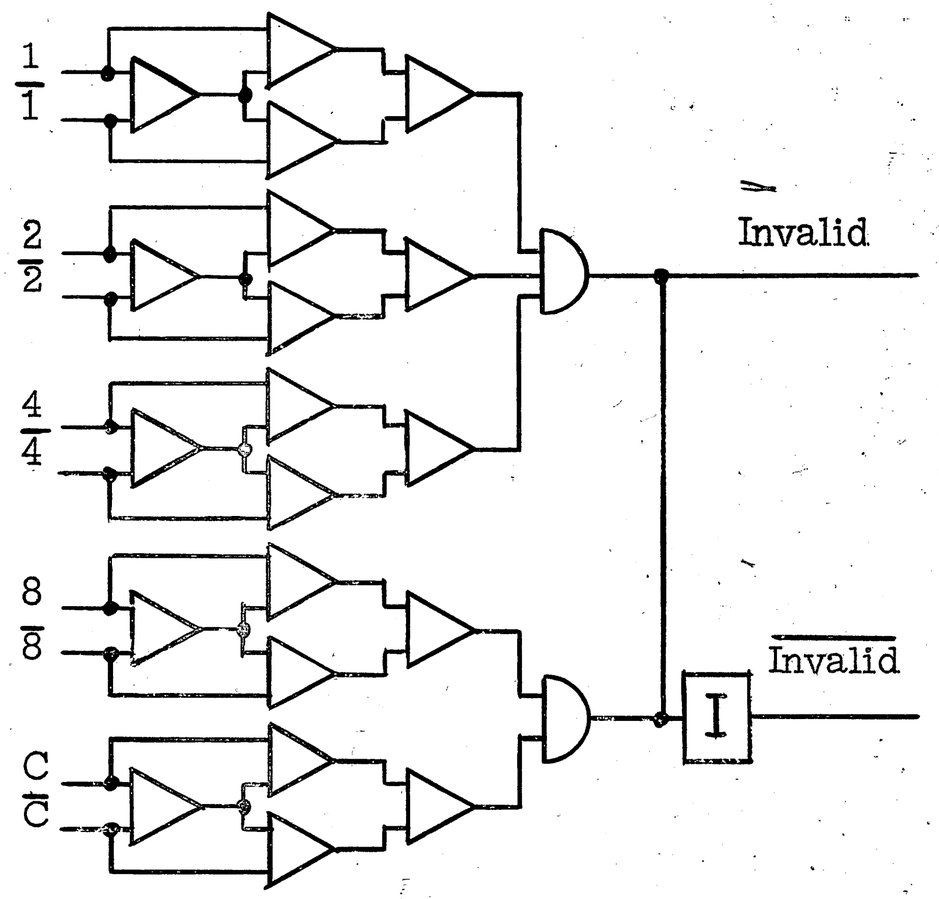
The check network consists of five exclusive or circuits which are "and"ed together to give a line which is labeled "-s valid." Each exclusive or circuit has a bit and a not bit for an input. Another set of exclusive or circuits are "or"ed together to give a "+s invalid" line. These lines are then combined according to the following expression to give an error signal.

$$\text{Error} = \text{Invalid} \cdot \text{Valid} + \text{Invalid} \cdot \overline{\text{Valid}} + \overline{\text{Invalid}} \cdot \overline{\text{Valid}}$$

The circuits are shown on the following pages. A data flow for the robot is also shown.

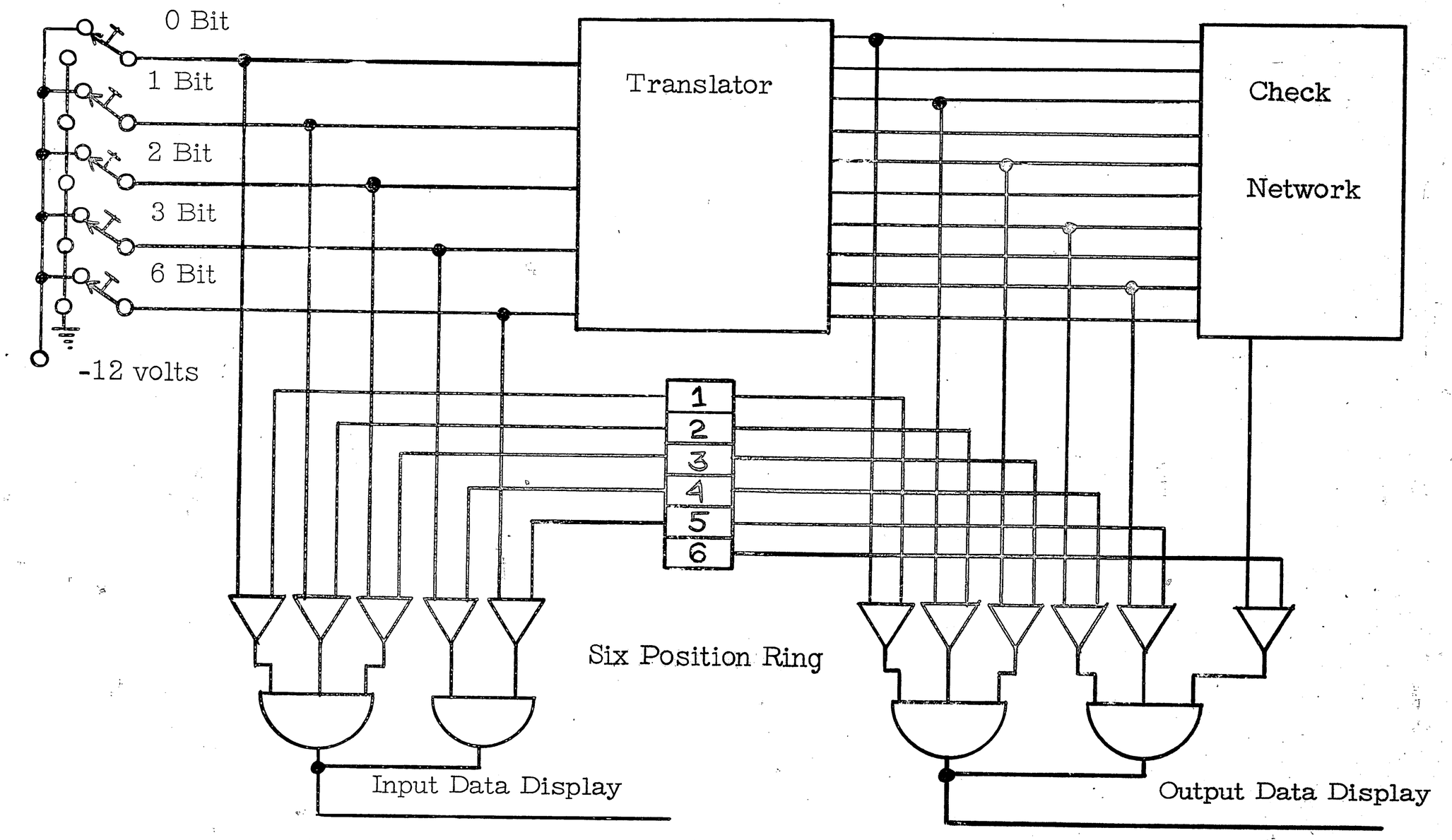
The following pictures were taken as the input data and output data were displayed on a Tektronix 545 Dual Trace Oscilloscope. The output data is shown for various combinations of input data. The upper trace on each picture is the input data with the bits in the following order -- 0 bit, 1 bit, 2 bit, 3 bit, 6 bit. The output data is shown as the lower trace with the bits occurring in the following order -- 1 bit, 2 bit, 4 bit, 8 bit, C bit, and error signal. The check network is shown in Figure 32; the test setup, Figure 33.



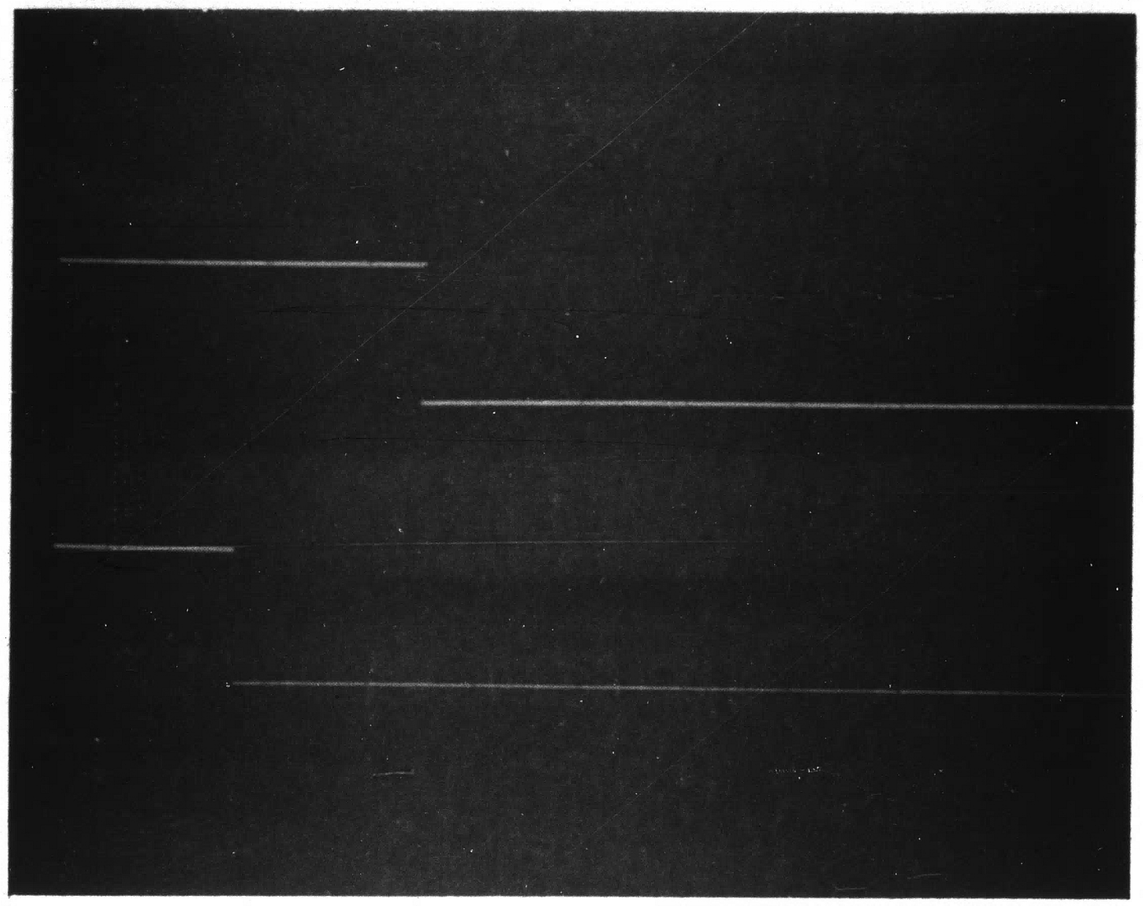


Fail safe Error Detection Network for Translator Output

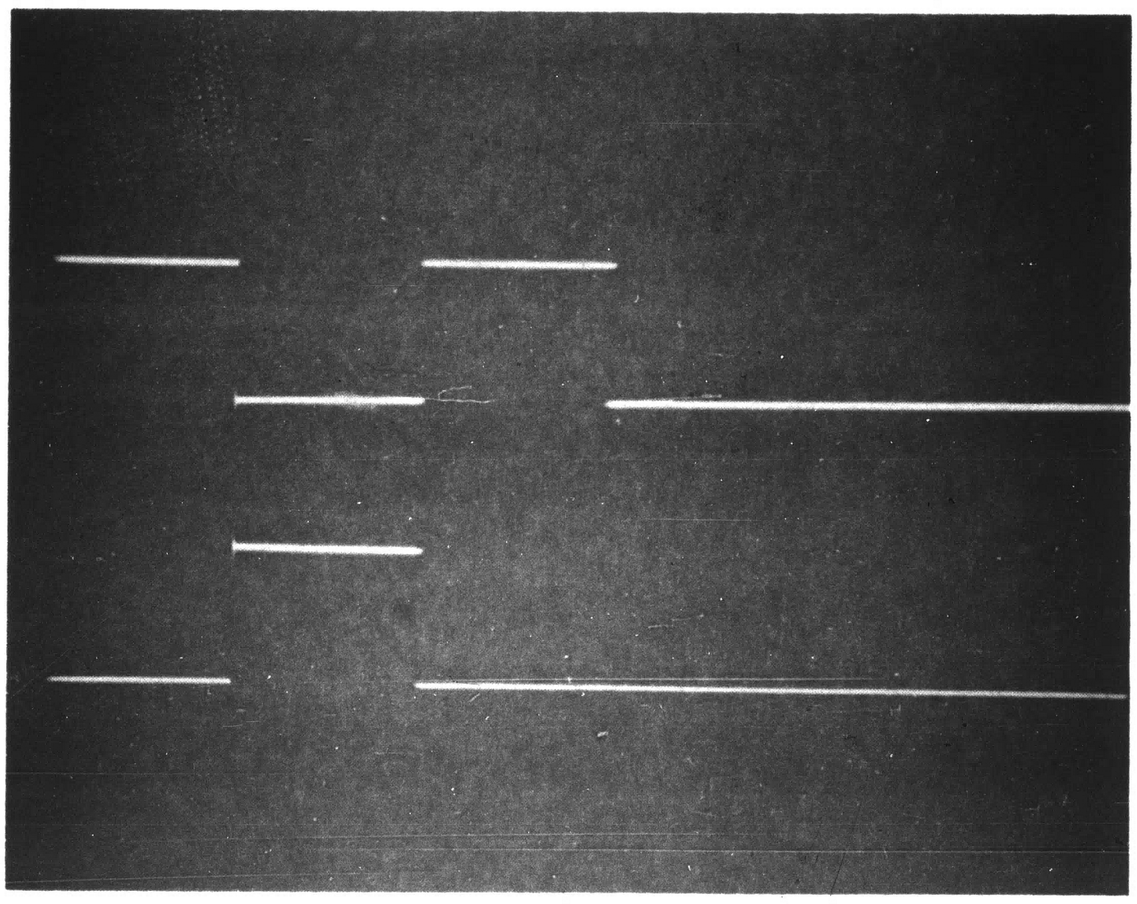
Figure 32



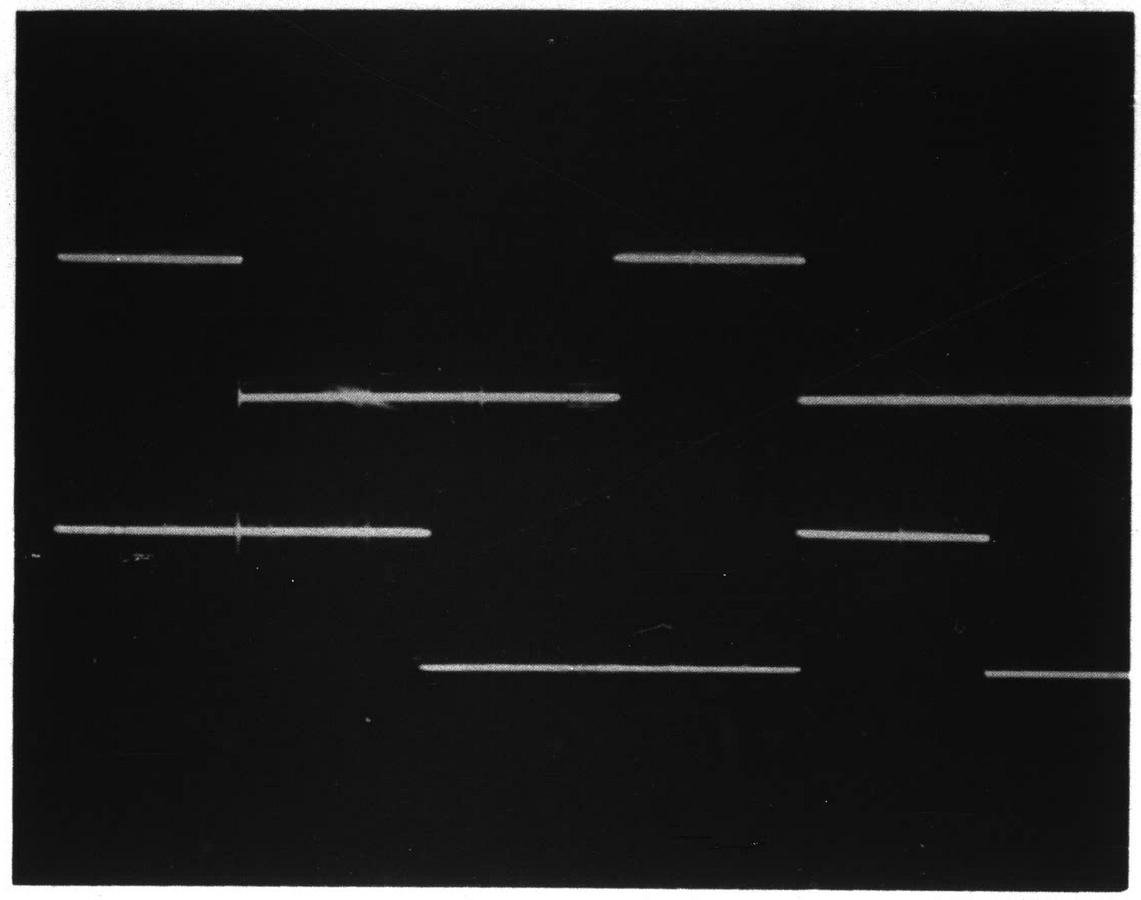
Test Robot Data Flow
Figure 33



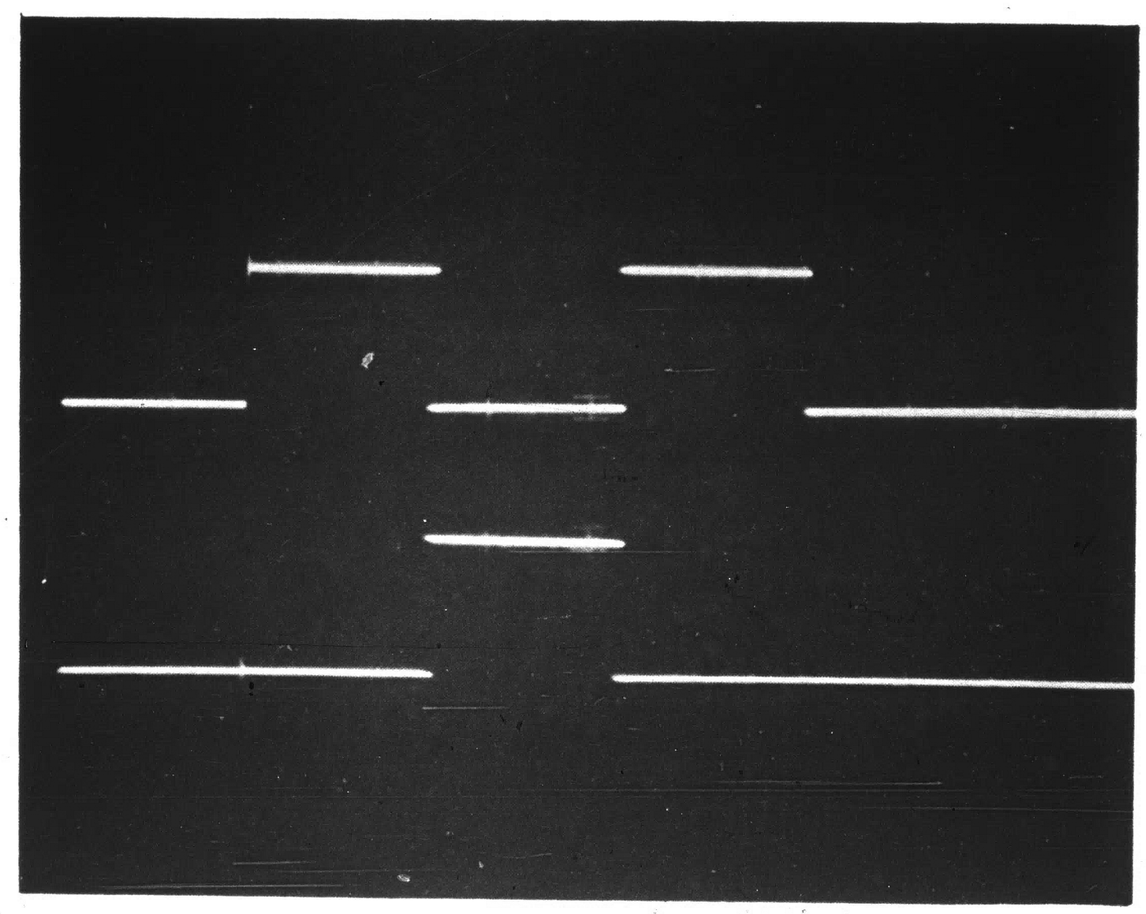
Input: 0 bit, 1 bit
Output: 1 bit
Figure 34



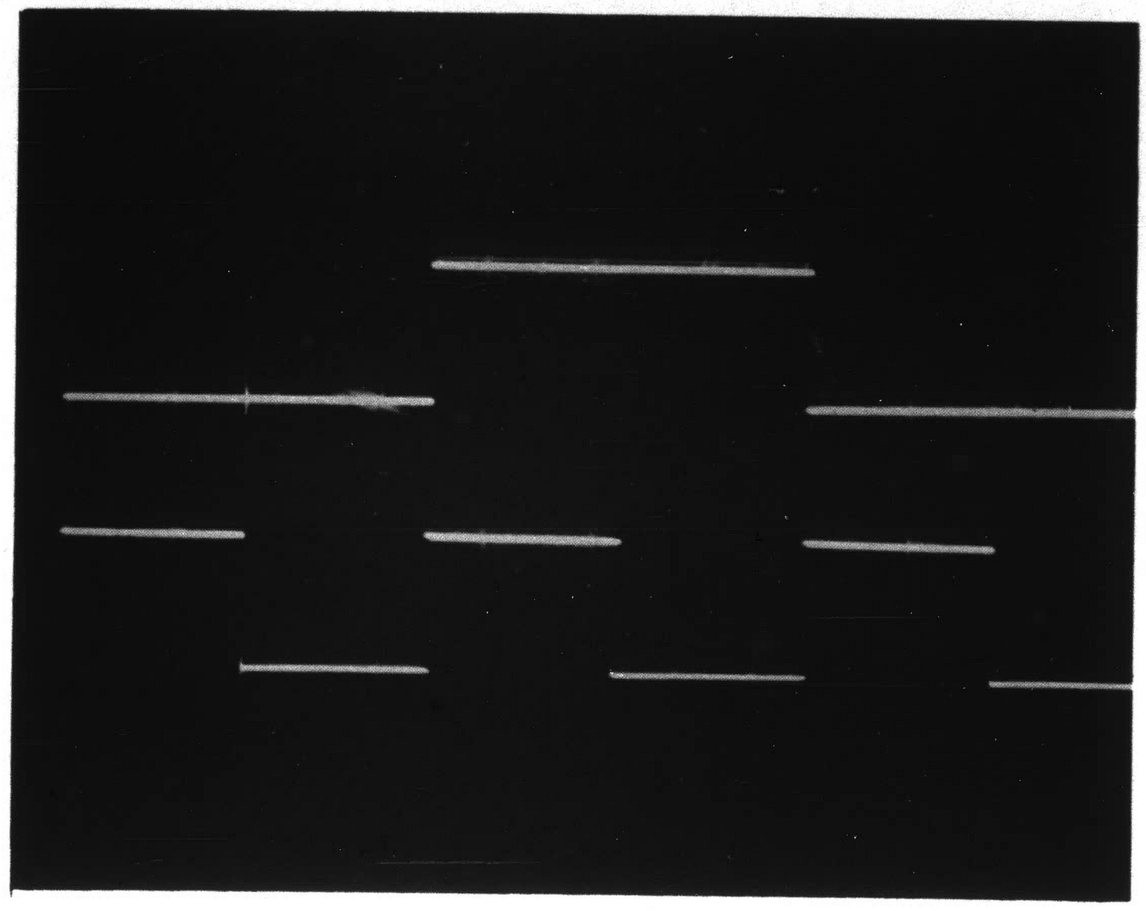
Input: 0 bit, 2 bit
Output: 2 bit
Figure 35



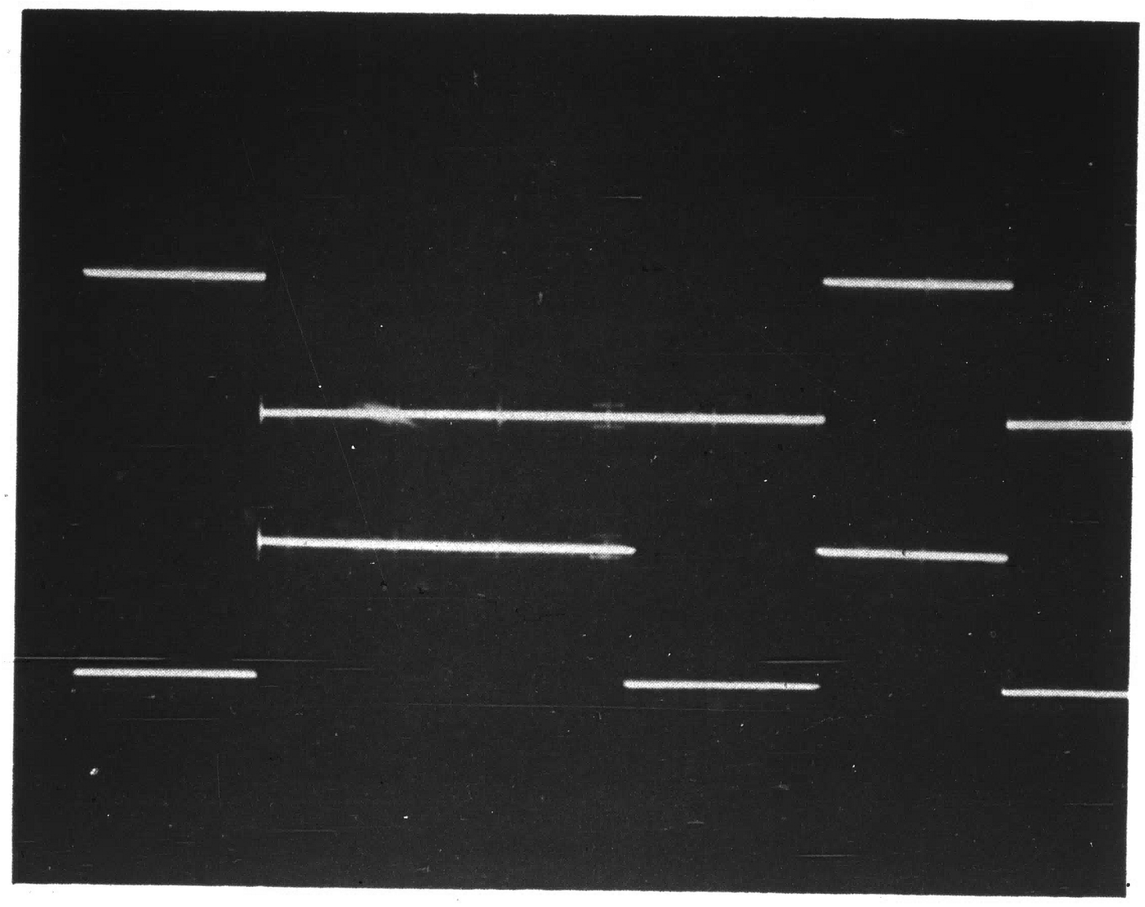
Input: 0 bit, 3 bit
Output: 1 bit, 2 bit, C bit
Figure 36



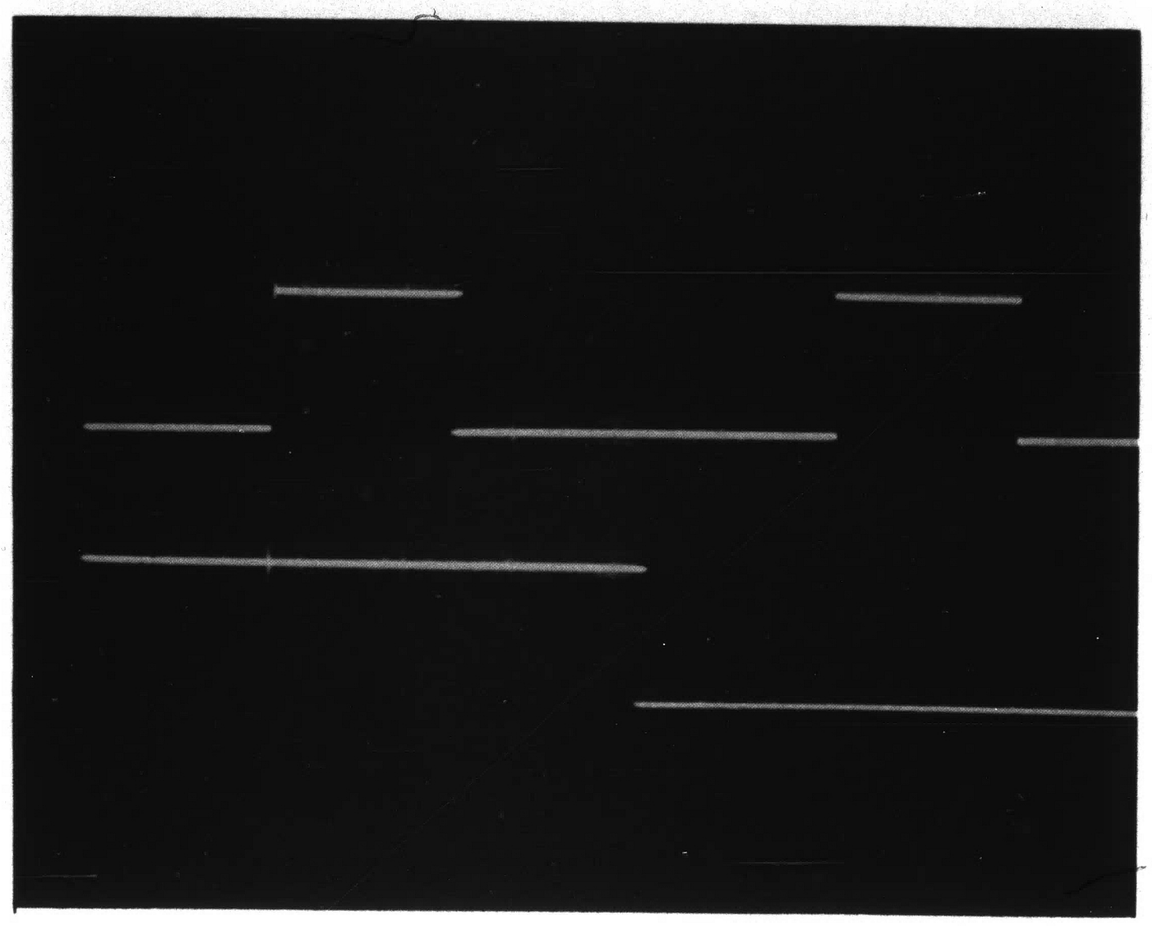
Input: 1 bit, 3 bit
Output: 4 bit
Figure 37



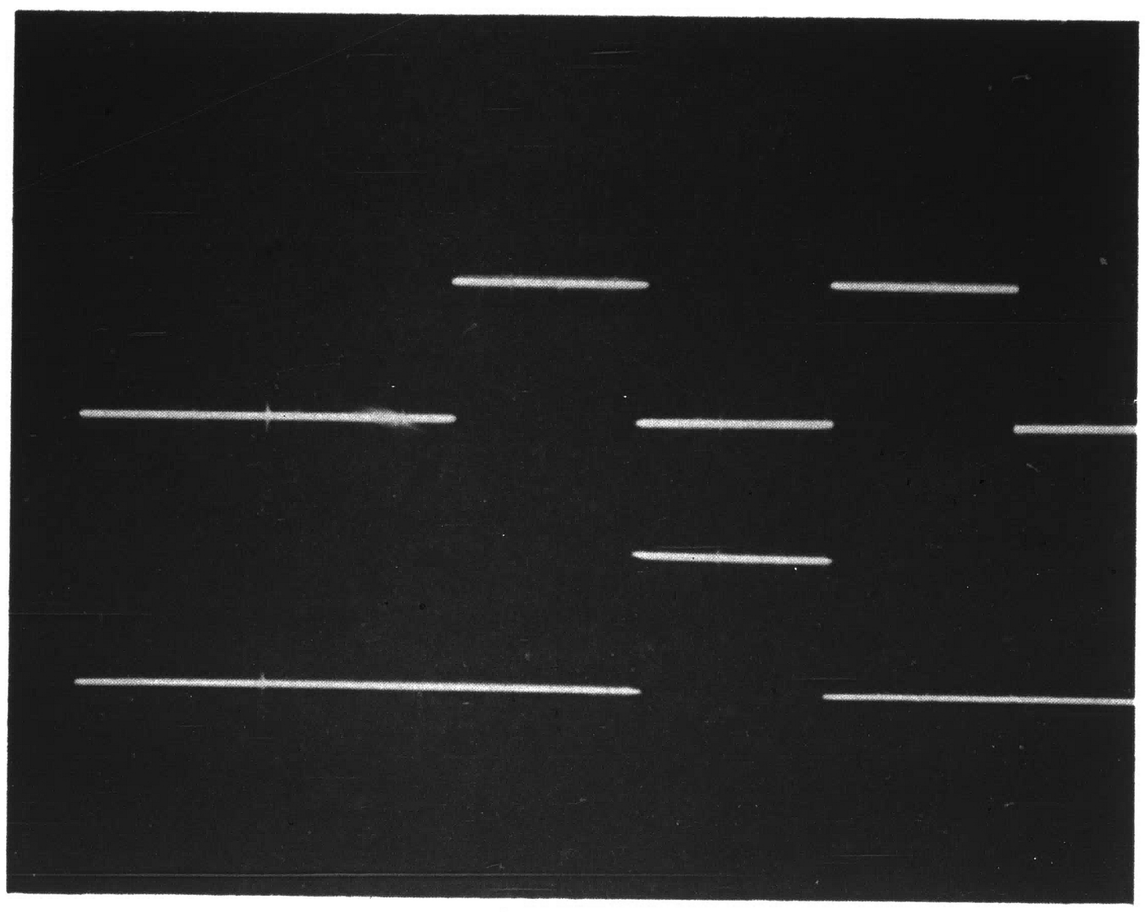
Input: 2 bit, 3 bit
Output: 1 bit, 4 bit, C bit
Figure 38



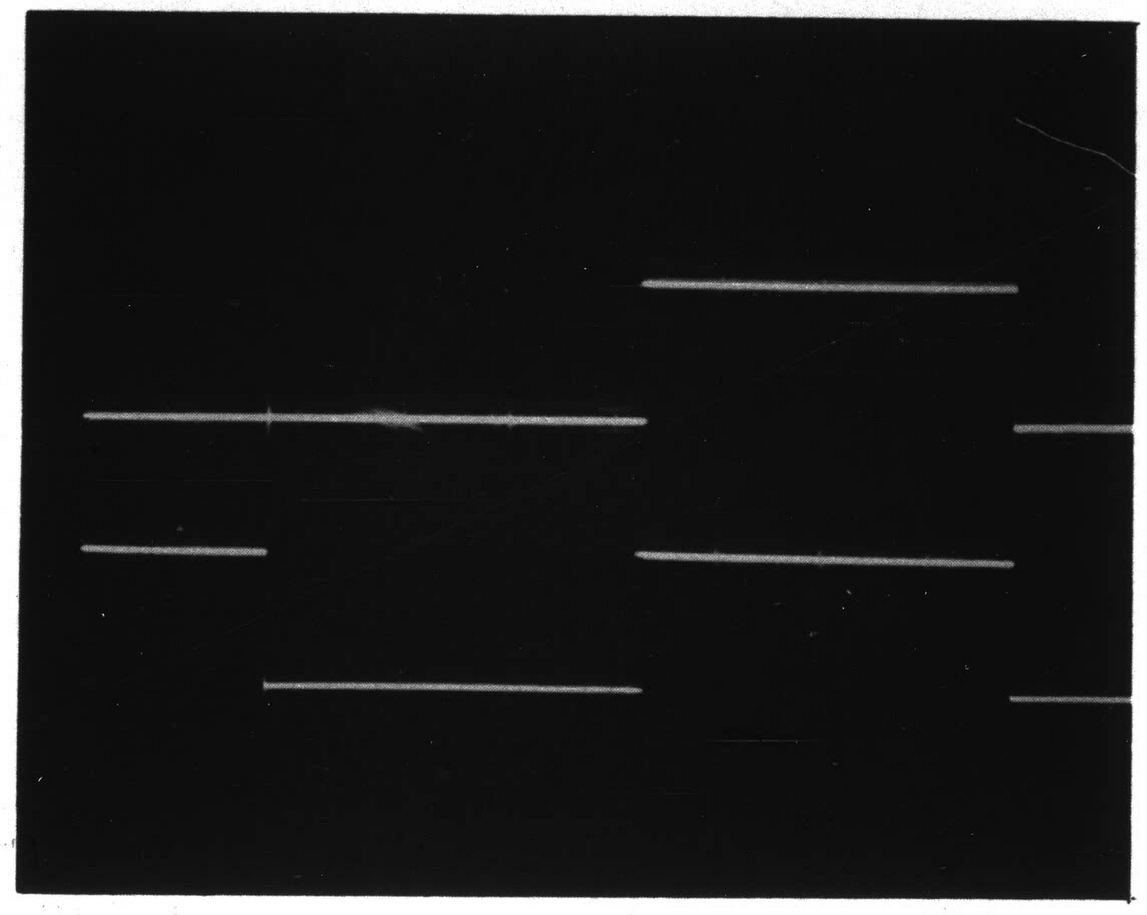
Input: 0 bit, 6 bit
Output: 2 bit, 4 bit, C bit
Figure 39



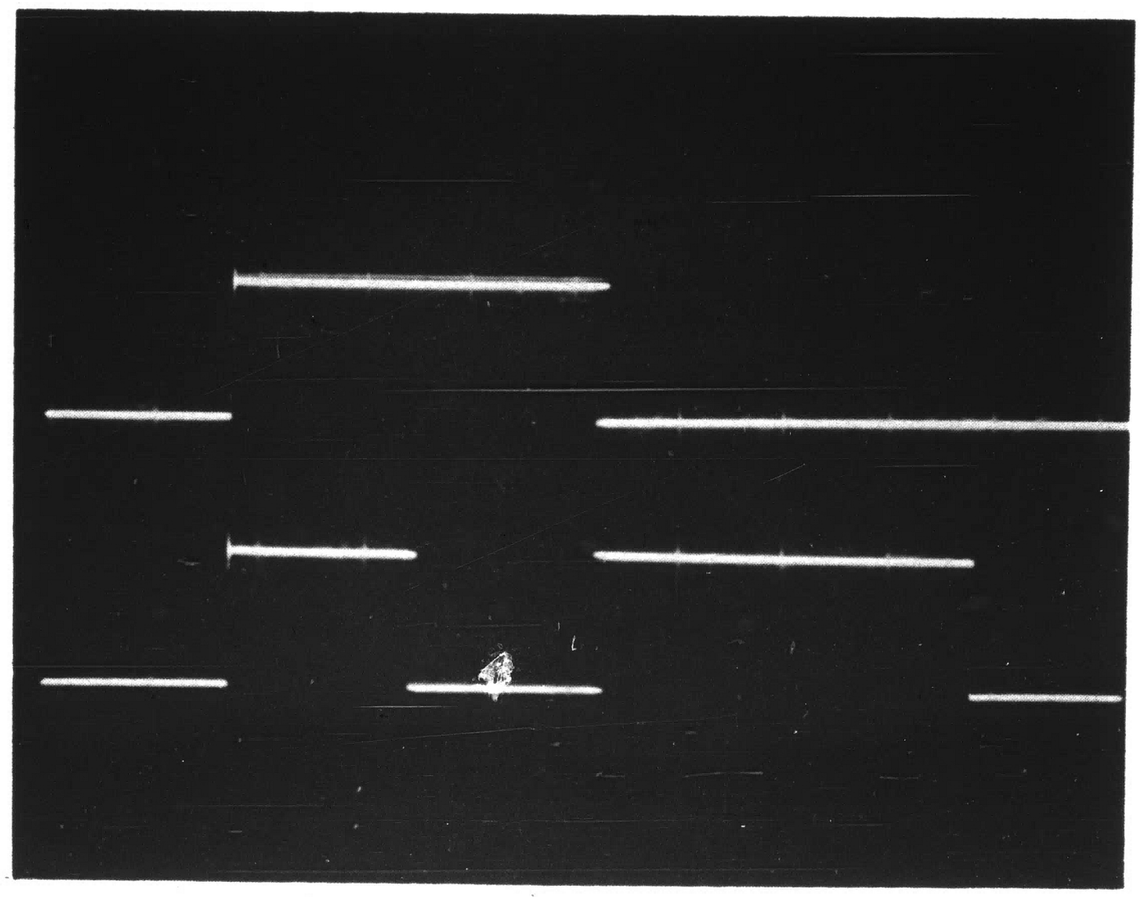
Input: 1 bit, 6 bit
Output: 1 bit, 2 bit, 4 bit
Figure 40



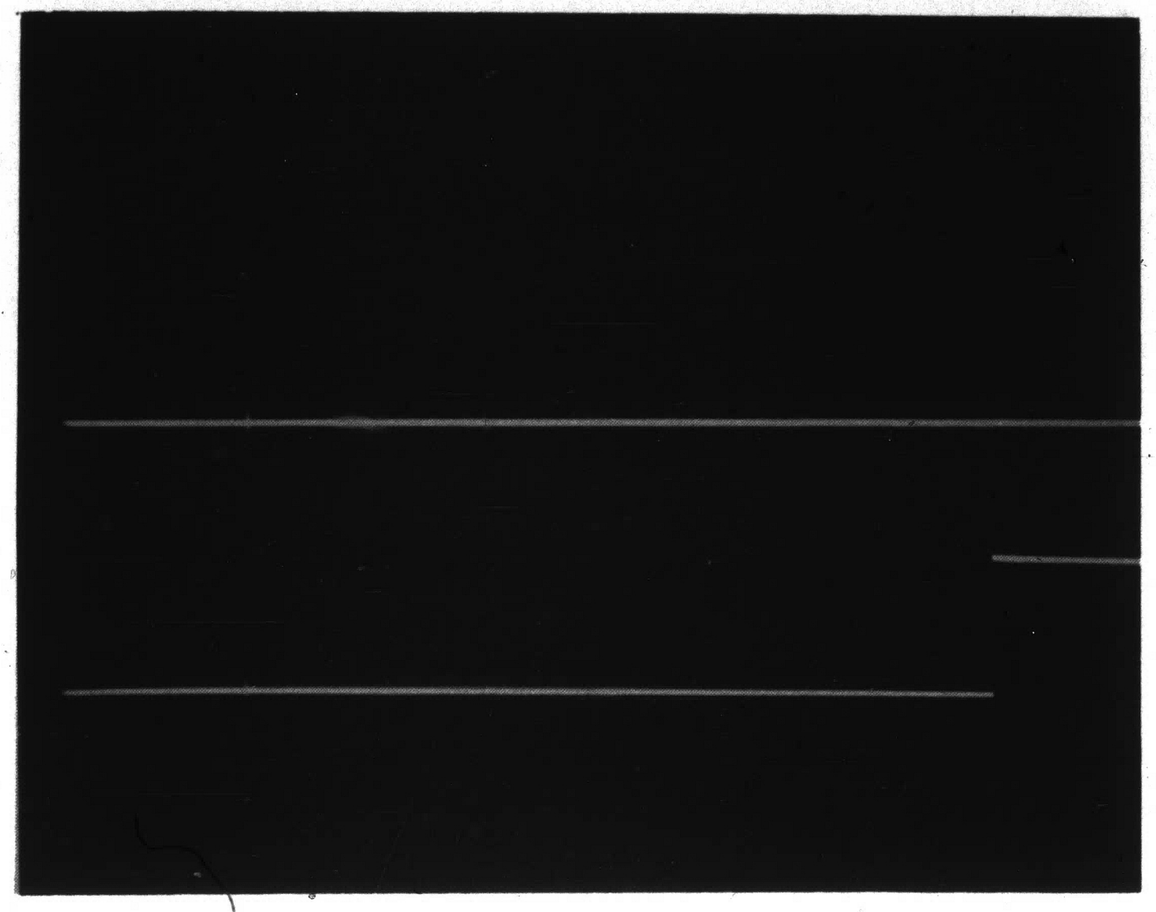
Input: 2 bit, 6 bit
Output: 8 bit
Figure 41



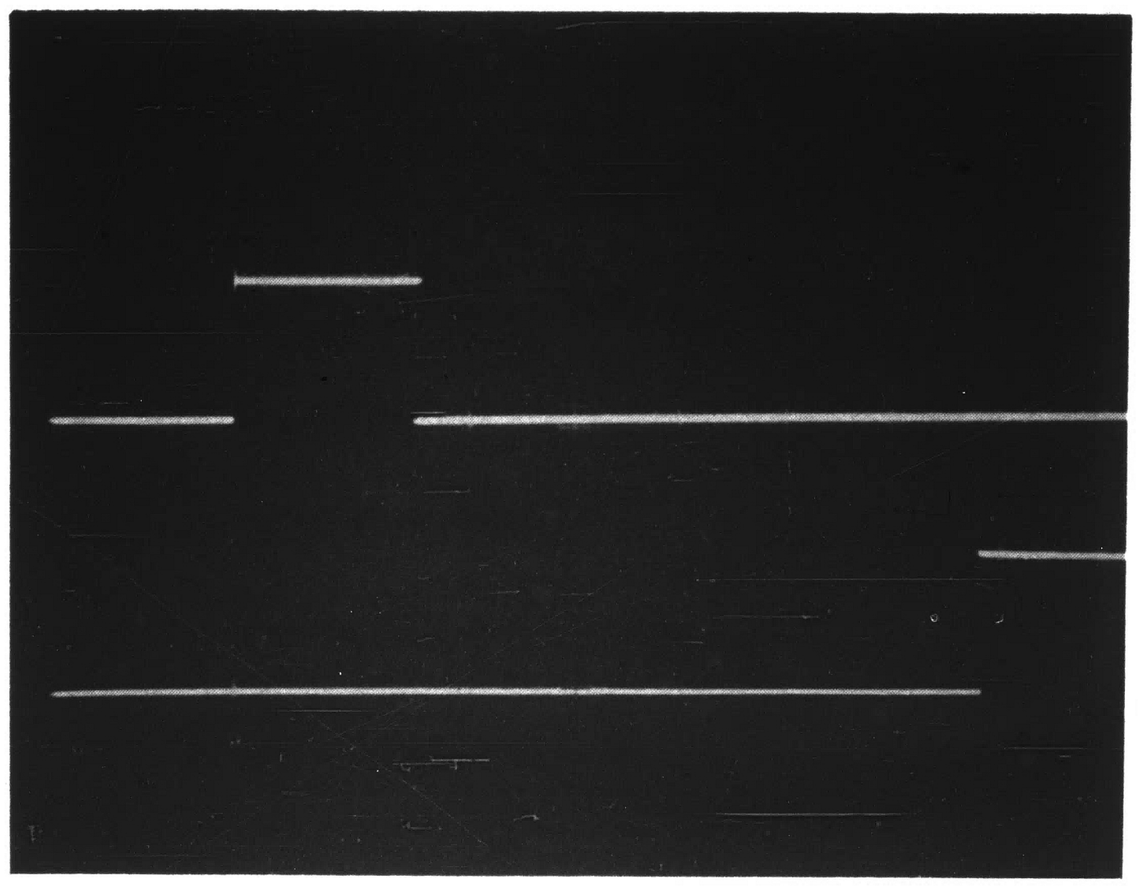
Input: 3 bit, 6 bit
Output: 1 bit, 8 bit, C bit
Figure 42



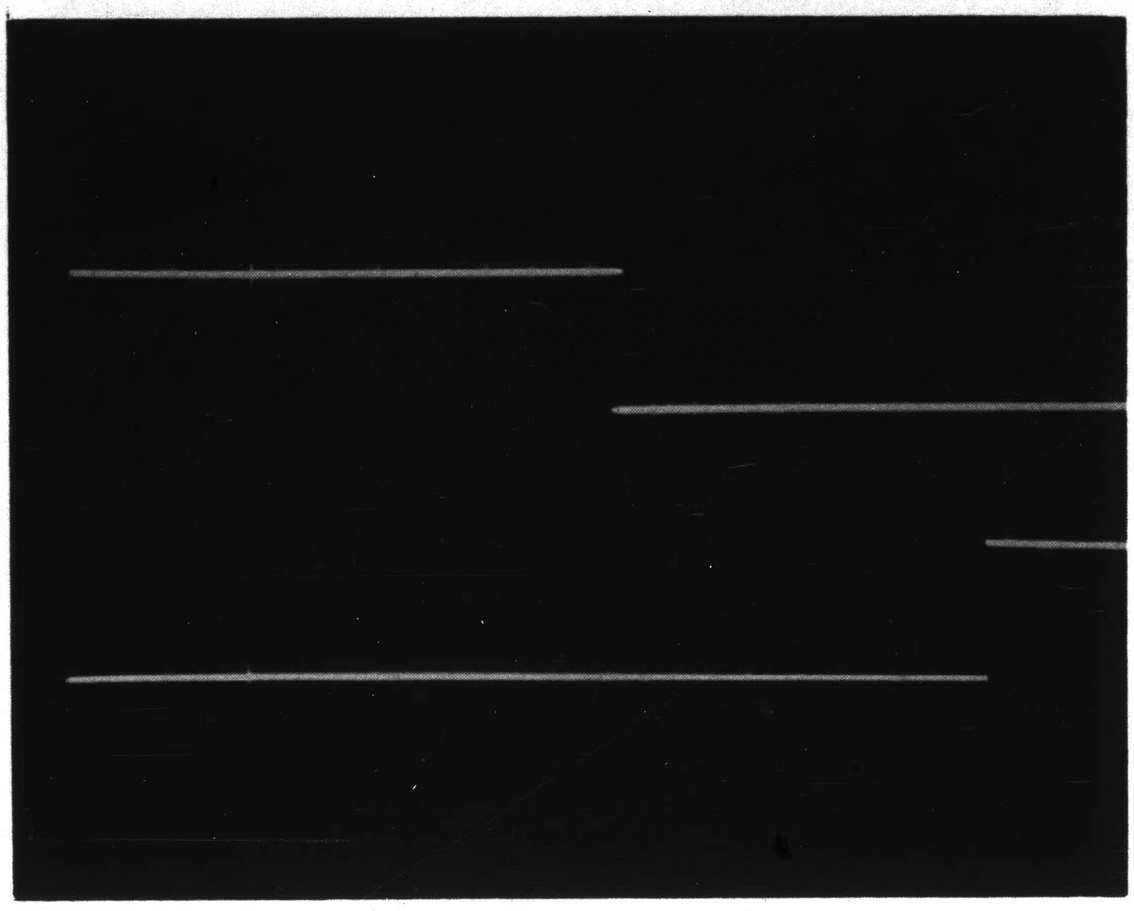
Input: 1 bit, 2 bit
Output: 2 bit, 8 bit, C bit
Figure 43



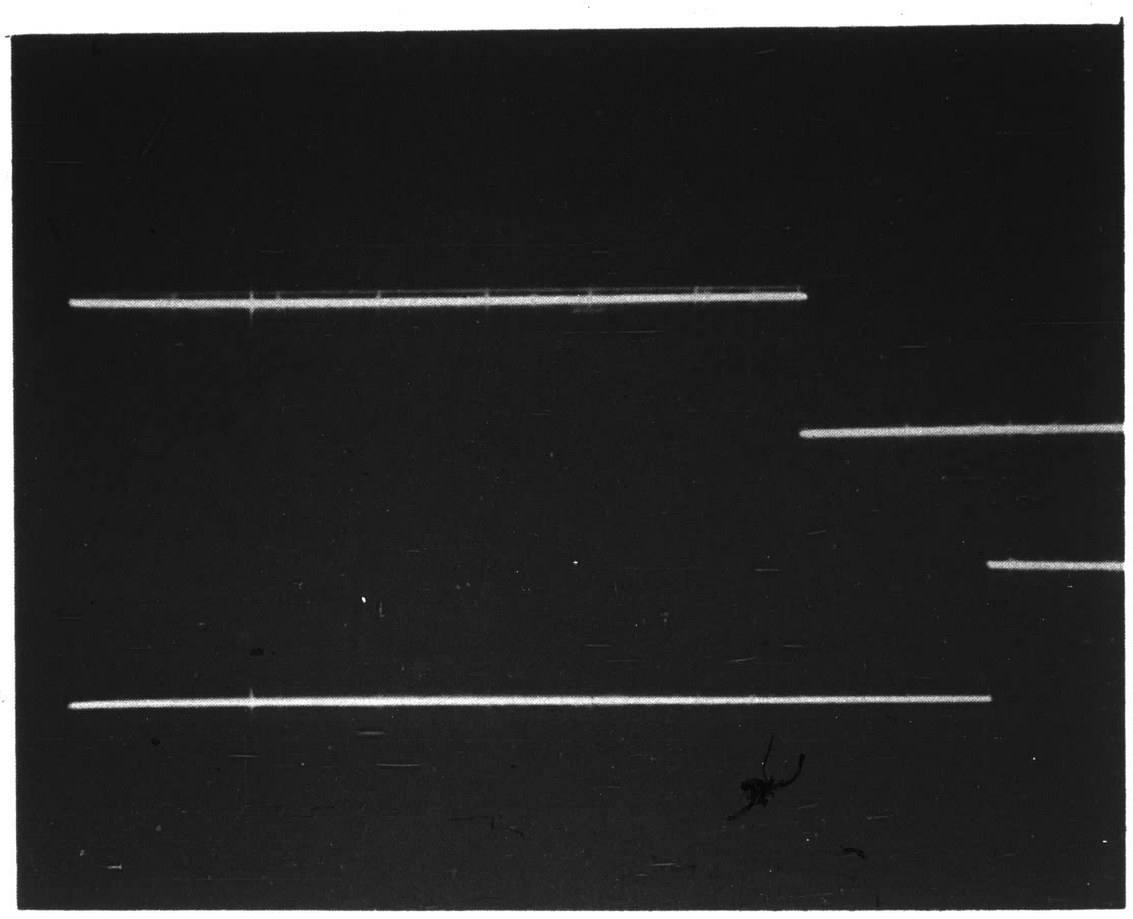
Input: No Bits
Output: No bits, Error Signal
Figure 44



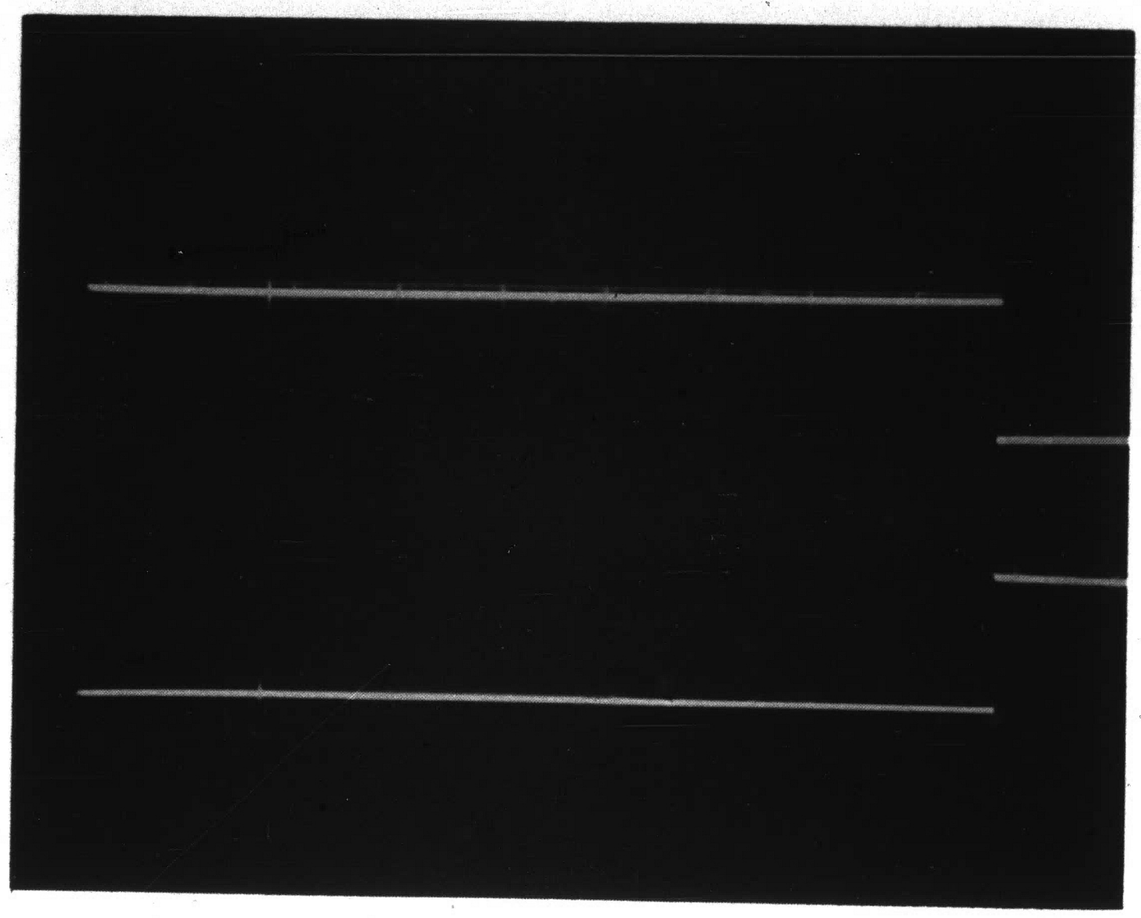
Input: 1 bit
Output: No bits, Error Signal
Figure 45



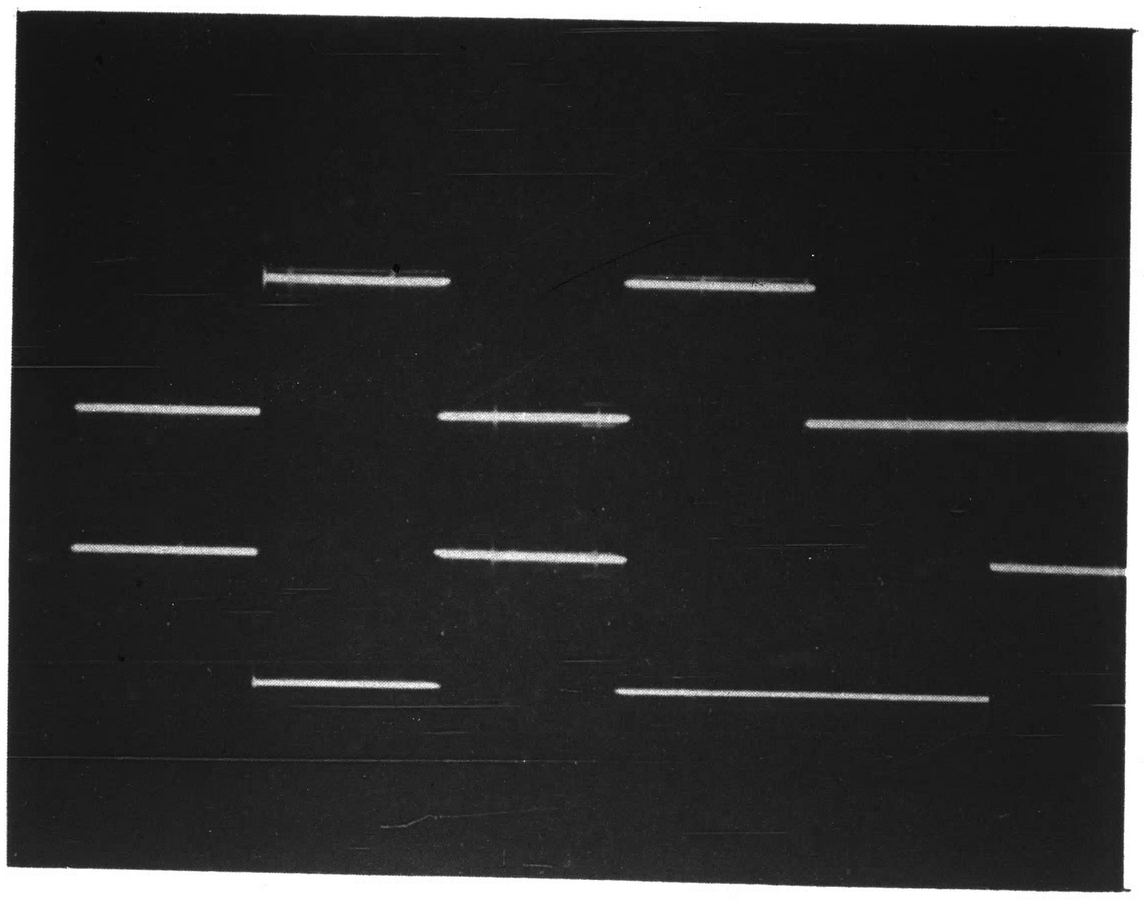
Input: 0 bit, 1 bit, 2 bit
Output: No bits, Error Signal
Figure 46



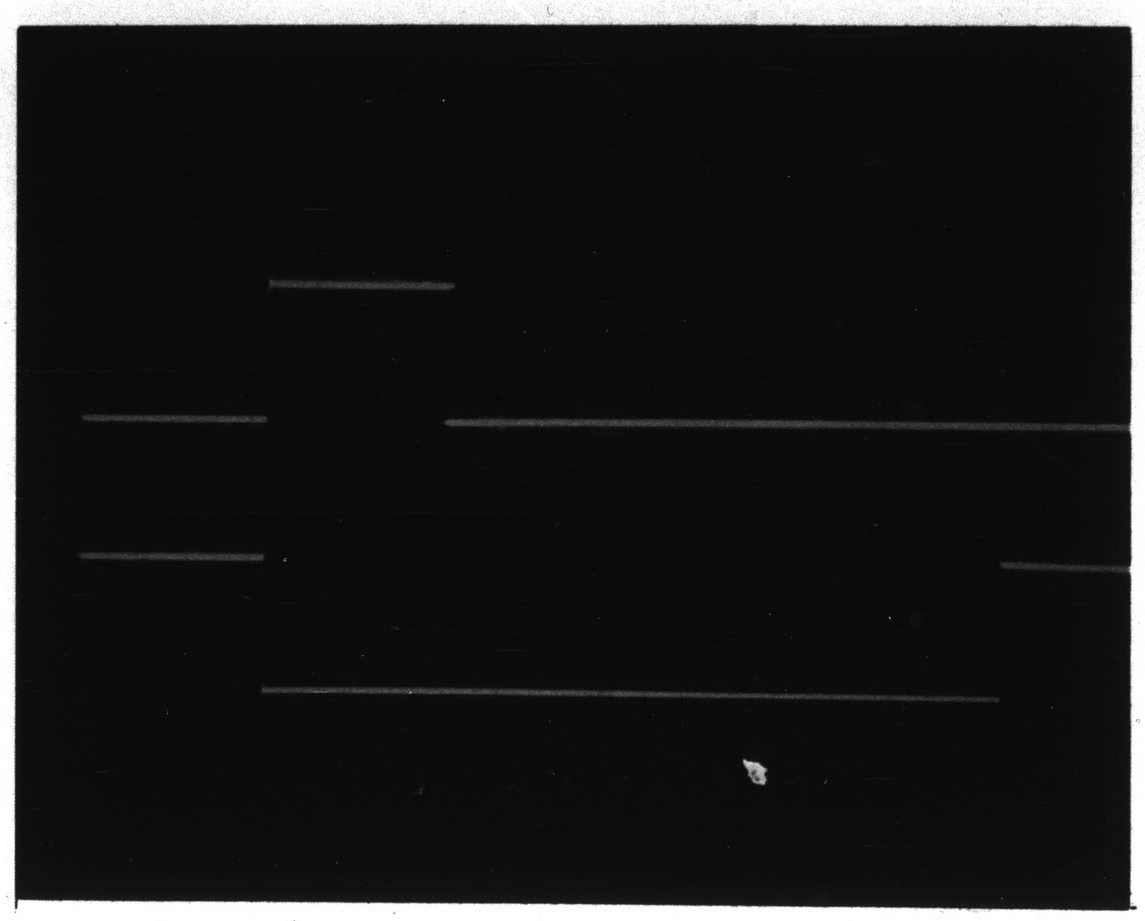
Input: 0 bit, 1 bit, 2 bit, 3 bit
Output: No bits, Error Signal
Figure 47



Input: All bits
Output: No bits, Error Signal
Figure 48



Input: 1 bit, 3 bit - Circuit Failure
Shorted Transistor in the "one" bit or circuit
Output: 1 bit, 4 bit, Error Signal
Figure 49



Input: 1 bit - Circuit Failure
Shorted Transistor in the no bit generator for the zero bit
Output: 1 bit, Error Signal
Figure 50

BIBLIOGRAPHY

Caldwell, Sameul H., Switching Circuits and Logical Design (New York, 1958).

Dunn, William J., "Switching Circuits" (Endicott, New York, 1958 - Notes from course of the same title).

Henle, R. A. and J. L. Walsh, "The Application of Transistors to Computers," Proceedings of the IRE, XLVI (1958), 1240-1254.

Richards, R. K., Arithmetic Operations in Digital Computers (Princeton, New Jersey, 1958).

Saxenmeyer, George J., "Computer Principles" (Endicott, New York, 1959 - Notes from course of the same title).

Suppes, Patrick, Introduction to Logic (New York, 1958).

VITA

The author was born on December 4, 1936, in Hazleton, Pennsylvania, the first son of Robert Edwin and Ruth Price Hughes. In June of 1954, he was graduated from Hazleton Senior High School.

In September of 1954, the author began his undergraduate work at Lehigh University which was completed in June, 1958, when he was graduated with highest honors and was awarded a Bachelor of Science Degree in Electrical Engineering. During his undergraduate years he served as president and secretary of the Delta Sigma Phi Fraternity and as a member of the Junior and Senior Class Cabinets. He is a member of Phi Eta Sigma, Pi Mu Epsilon, Eta Kappa Nu, Tau Beta Pi, and Phi Beta Kappa. He also received Freshman and Sophomore Honors, a Sperry Gyrsocope Company Scholarship, and was on the Dean's list for eight semesters.

Having been awarded an International Telephone and Telegraph Company Fellowship, the author began his graduate study in September, 1958. Since July, 1959, he has been employed at Endicott, New York, by the International Business Machines Corporation.

He was married on September 6, 1958, to the former Carol Anne Lapinsky and is now the father of two children--Cheryl Lynn, aged three, and Craig David, aged one.